

# Quantitative $\mu$ -calculus and CTL defined over constraint semirings<sup>1</sup>

Alberto Lluch-Lafuente and Ugo Montanari

*Dipartimento di Informatica  
Università di Pisa*

---

## Abstract

Model checking and temporal logics are boolean. The answer to the model checking question *does a system satisfy a property?* is either *true* or *false*, and properties expressed in temporal logics are defined over boolean propositions. While this classic approach is enough to specify and verify boolean temporal properties, it does not allow to reason about quantitative aspects of systems. Some quantitative extensions of temporal logics has been already proposed, especially in the context of probabilistic systems. They allow to answer questions like *with which probability does a system satisfy a property?*

We present a generalization of two well-known temporal logics: CTL and the  $\mu$ -calculus. Both extensions are defined over c-semirings, an algebraic structure that captures quantitative aspects like quality of service or soft constraints. Basically, a c-semiring consists of a domain, an additive operation and a multiplicative operation, which satisfy some properties. We present the semantics of the extended logics over transition systems, where a formula is interpreted as a mapping from the set of states to the domain of the c-semiring, and show that the usual connection between CTL and  $\mu$ -calculus does not hold in general. In addition, we reason about the complexity of computing the logics and illustrate some applications of our framework, including boolean model checking.

*Key words:* Semirings, Temporal Logics, QoS, Quantitative Model Checking.

---

## 1 Introduction

Model Checking [10] is a formal automated method for system verification consisting of three main steps: modeling the system, modeling the properties

---

*Email address:* {lafuente,ugo}@di.unipi.it (Alberto Lluch-Lafuente and Ugo Montanari).

<sup>1</sup> Work supported by the European Research Training Network SEGRAVIS, Project MIUR/CNR SP4 and the European FET Project Profundis.

of the system and checking whether the properties hold in the system. Usually, systems are represented using formalisms such as automata, labeled transition systems and Kripke structures with a finite number of states, where boolean propositions are associated to states and transitions. Properties of the system are often represented using temporal logics, which combine modal operators with boolean connectives and propositions. The verification step returns a boolean answer: either *yes* (the system satisfies the specification) or *false* (the system violates the specification).

**Motivation.** While this approach is sufficient to reason about *qualitative* aspects of systems, it is clearly not sufficient to reason about *quantitative* aspects. Hence, approaches have been proposed for the analysis and verification of quantitative systems. Among others, we cite durational systems [26], probabilistic systems [18,21] and timed systems [1]. Formalisms to represent systems and properties are equipped with quantitative aspects such as time or probabilities. The verification question can be boolean (based on thresholds) or purely quantitative. For example, when reasoning about a probabilistic system one can ask if a property holds with a probability higher than  $p$ , or which is the probability that a property holds.

A similar situation exists in the domain of Constraint Satisfaction Problems (CSP). There, classical problems consider boolean constraints only: either a constraint is present or it is not. Some CSP problems, however, cannot be properly formulated using this crisp interpretation of constraints. Several approaches exist to remedy this, which basically associate non-boolean values to constraints, like probabilities, costs and sets. Such constraints are called soft constraints and the corresponding problems are soft CSP, like fuzzy CSP, probabilistic CSP and weighted CSP, among others.

A general framework for soft CSP is proposed in [4–6]. The key of the approach is an algebraic structure called *constraint semiring* (c-semiring for short). A c-semiring consists of a domain and two operations, which are called additive and multiplicative operations. The basic idea is that the domain is used to represent the values associated to constraints (booleans, probabilities, etc.), the additive operation is used to project constraints and the multiplicative operation is used to combine constraints. The framework captures most of the common soft CSP problems, such that the results stated for the framework can be applied to concrete instances, for example, as hints for the applicability of special solution methods for a concrete instance. More interestingly, c-semiring constraint methods have a unique advantage when problems with multiple criteria or multiple metrics must be tackled. In fact, it turns out that Cartesian products, exponentials and power constructions of c-semirings are c-semirings. Thus the same concepts and algorithms can be applied again and again.

**Goals.** The purpose of our research is to define a general framework for quantitative verification based on c-semirings, with a special focus on *Quality of Service* (QoS) properties. In this paper we present our first step. We have ex-

tended two well-known temporal logics,  $\mu$ -calculus and CTL, where boolean propositions and connectives are replaced by c-semiring propositions and operations, while temporal operators are substituted by operators on sequences of c-semiring values. We present a semantics over transition systems. While in boolean model checking the interpretation of a formula can be seen as a mapping from the set of states to the boolean domain, which implicitly represents the subset of states satisfying the formula, we interpret formulae as mappings from the set of system states to the domain of the c-semiring.

As in the general framework for soft CSP [6] we analyze and present results for a specific kind of c-semirings called distributive c-semirings. Distributive c-semirings cover a significant part of c-semirings and have interesting properties that make them easier to be handled. Similarly, we also distinguish fragments of our logics, when presenting our results. The main idea behind such distinctions is to offer, for specific cases of our framework, specific methods. We shall see, for instance, that the expressive power of  $\mu$ -calculus and CTL is comparable for distributive c-semirings and that some fragments of our logics are easier to verify.

While we expect our framework to capture many problems, one of our main interests is to use it as a formalism to analyze QoS properties. QoS are measures of the non functional properties of systems. Bandwidth, delay and jitter are typical examples of network QoS properties, while application-level QoS includes, among others, price and access rights. Different kinds of semirings have been proposed to model costs, for instance in the algebraic path problem [24], but c-semirings are sufficient to capture most of the significant QoS attributes used in practice. As a matter of fact, c-semirings has been used in Kaos [22], a calculus for programmable QoS, as a formalism for representing QoS properties of WAN applications.

An example where reasoning about QoS is a crucial aspect are *Service Overlay Networks* [14] (SONs) which have gained major attention during recent years as a flexible mechanism to manage the complexity in the creation and deployment of network services with certain Quality of Services (QoS) assurances. A challenging problem is to provide formal machineries that support verification of both behavioral and QoS properties of SON in an integrated way.

As a simple example consider a system involving various agents and resources. Suppose we are interested in observing not only whether a certain agent is using a certain resource, but the QoS level of that usage. While in a boolean setting one is interested in properties like *will the agent ever use the resource?*, in the proposed scenario one could be interested in expressing properties like *which is the best QoS level (of the usage of the resource by the agent) ever achieved?*. Note that different resources or agents might involve different QoS attributes. For instance, the usage of a resource  $A$  might be involve a price to pay and a performance rate, while the usage of a resource  $B$  might involve access rights. Thus, we need a general concept of QoS that captures many of

the typically used QoS attributes offering a framework that abstracts from concrete attributes and provides general mechanisms, including the property of combining several attributes. We believe that c-semirings are a suitable formalism for this purpose.

**Related work.** Our approach is not the first attempt to define a framework for quantitative verification. For instance, the algebraic  $\mu$ -calculus [2] extends the classical  $\mu$ -calculus with arithmetic expressions. While our approach basically substitutes boolean connectives and quantifiers with c-semiring operations, the algebraic  $\mu$ -calculus substitutes boolean connectives and quantifiers with arithmetic operations and quantifiers. However, contrary to our extension of the  $\mu$ -calculus, the algebraic  $\mu$ -calculus substitutes the fixpoint operators with a limit operator, which is more general and indeed shown to turn into least and greatest fix-point operators.

The approach is very general and embeds a wide range of problems. The algebraic  $\mu$ -calculus captures approaches that are captured by our framework too. An example are graph theoretical problems like the shortest path problem. On the other hand, the algebraic  $\mu$ -calculus generalizes probabilistic approaches that our approach does not capture, like the verification of probabilistic systems with PCTL [17], a probabilistic extension of CTL. It is an open question whether the algebraic  $\mu$ -calculus captures c-semiring  $\mu$ -calculus.

One of the drawbacks of the algebraic  $\mu$ -calculus in comparison to our work is that the approach is too general such that most of the obtained results seem to be of little practical use. Our approach, instead, presents different results and specifies them for concrete cases of the approach as explained above. In addition, representing quantitative aspects like Quality of Service issues using an arithmetic algebra is rather unnatural, while c-semiring are very suitable for that purpose. Finally, a path logic like c-semiring CTL is definitively easier to understand and use than a fixpoint logic.

Our work is also inspired by previous research on quantitative analysis and verification of probabilistic systems. Contrary to approaches for the qualitative verification of probabilistic systems like PCTL [17], where the evaluation of formulae is boolean, in quantitative approaches the evaluation of formulae are probabilistic values. In [18], for instance, a probabilistic extension of the  $\mu$ -calculus is proposed. The syntax of the logic is the exactly that of the propositional  $\mu$ -calculus. The semantics is defined over probabilistic transition systems, where three alternative probabilistic semantics for disjunction and conjunction are presented. Since fixpoint iteration is infeasible, they propose an alternative approach to evaluate a significant fragment of their logic, which basically consists on reducing the problem of fixpoint computation to an (equivalent) optimization problem in linear programming.

The implicit algebraic structure used in [18] is not a c-semiring. Thus, our approach does not capture that approach, but, on the other hand, it is clear

our work can be applied to represent problems that the probabilistic approach of [18] cannot represent, like shortest path problems, for instance.

The work described in [12] defines two quantitative extensions of the  $\mu$ -calculus: a probabilistic one, where disjunction and conjunction of probabilities are given a fuzzy (min/max) interpretation, and a discounted one, where events are weighted according to their distance to the present. Their semantics is defined over games, a formalism that generalizes, among others, boolean and probabilistic transition systems. The same authors propose a discounted version of CTL in [13], where two semantics are given: the path and fixpoints semantics. While in boolean model checking both semantics are equivalent, allowing the use of fixpoint iterations as algorithms for CTL, they differ when discounted CTL is interpreted over probabilistic systems. Specific algorithms for each semantics are thus proposed.

Despite the fuzzy interpretation of probabilities can be captured with a c-semiring properly, the discounting factor, however, and the fact that the semantics are interpreted over games, while our work restricts to transition systems, are enough to make our work and the two just mentioned approaches [12,13] incomparable. We shall see, however, that our approach is able to express discounted model checking problems for transition systems.

Some probabilistic approaches [18] provide boolean notions of equivalence relations like bisimulation, while others [12] define quantitative relations. While in the former, two states are either related or not, in the latter two states are related by a probability value, which represents a sort of distance between the two states. The value resulting from the evaluation of a probabilistic formula is shown to depend on their distance. We present, however, equivalence and preorder relations that are still boolean. The main reason for this is that it is not possible to define a meaningful notion of distance in c-semirings.

Our treatment of probabilities is simpler in comparison with probabilistic approaches like the ones just mentioned [12,13,18]. This should not be a surprise, since probabilistic approaches focus on probabilities, while for our QoS perspective, probabilities are just one aspect. Hence, we do not propose our approach when the only quantitative aspect of the systems being analyzed are probabilities. Instead, we propose our work to be a suitable framework when quantitative aspects are of QoS attributes of different natures, all properly captured by c-semirings.

Another closely related work is multi-valued CTL [9], where CTL is defined over a quasi-boolean algebra. Briefly, a quasi-boolean algebra is a finite distributive lattice with a negation operator. The resulting logic is suitable for analyzing models with uncertainty and inconsistency. As we shall see c-semirings are more general than quasi-boolean algebras. Therefore, our work subsumes a great part of this approach. Multi-valued CTL, however, has some issues that our approach is not able to capture, like the fact that formulae might be

interpreted over multi-valued transition systems.

In sum, the main difference of our approach with respect to other works on quantitative verification of quantitative systems is that we do not focus in one or two concrete dimensions, like time or probabilities. Rather, we focus on multiple aspects. Therefore, we base our work on a general algebraic structure, which offers the possibility of combining different dimensions. Indeed, as we shall see c-semirings can be combined in different ways, such that the combination is still a c-semiring.

**Contribution.** The novelty of our approach is that our logics are defined over c-semirings, generalizing thus approaches based on quasi-boolean algebras and fuzzy probabilities, for instance. We define syntax and semantics of two logics: c-semiring CTL (c-CTL) and c-semiring  $\mu$ -calculus (c- $L_\mu$ ). We compare the semantics of c-CTL with an alternative semantics based on the fixpoint semantics of traditional CTL, shown to be equivalent to the usual path semantics of CTL. In our case this equivalence does not hold in general. In addition to the results in our previous work [19] we further refine the relation between both semantics and show that the usual notions of system equivalences and preorders like bisimulation or simulation are only suitable for some fragments of our logics. We show that in some particular c-semirings, which are actually distributive lattices, the model checking problem is decidable and give an upper bound on the complexity. For the general case, we show how the problem of computing formulae of some fragments can be reduced to well-known graph problems. These results are almost novel with respect to our previous work [19] where we just presented algorithms for model checking c-CTL in particular c-semirings.

**Structure of the paper.** This paper is structured as follows. Section 2 gives the necessary background on c-semirings and transition systems. Section 3 describes syntax and semantics of the extended logics. Section 4 is on the computation of our logics. The next section deals with equivalence and pre-order relations. Section 6 illustrates various applications of our framework. The last section concludes the paper and outlines current and future work.

## 2 Preliminaries

Our logics are defined over the domain of a c-semiring, which is a tuple  $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$  such that:

- $A$  is a set;
- $\mathbf{0}$  and  $\mathbf{1}$  are elements of  $A$ ;
- $+$  :  $2^A \rightarrow A$  is defined over (possibly infinite) sets of elements of  $A$  as

follows<sup>2</sup>:  $\sum\{a\} = a$ ,  $\sum\emptyset = \mathbf{0}$ ,  $\sum A = \mathbf{1}$  and  $\sum(\cup A_i) = \sum\{\sum A_i\}$ , for  $A_i \subseteq A$ ,  $i \geq 0$ ;

- $\times : A \times A \rightarrow A$  is a binary associative, commutative operation that distributes over  $+$ , has  $\mathbf{1}$  as its unit element and  $\mathbf{0}$  as its absorbing element.

The fact that  $+$  is defined over sets of elements, automatically makes such an operation associative, commutative and idempotent. Moreover, one can show that it has  $\mathbf{0}$  as unit element and  $\mathbf{1}$  as absorbing element [6]. Given a sequence  $a_0, a_1 \dots$  of elements of  $A$  we write  $\sum_{i \geq 0} a_i$  rather than  $\sum_{\{a_0, a_1, \dots\}}$ . Given a finite sequence  $a_0, \dots, a_n$  we abbreviate  $a_0 \times \dots \times a_n$  with  $\prod_{0 \leq i \leq n} a_i$ . In the rest of the paper we assume that  $\prod$  is defined over infinite sequences too. Most of the c-semirings used in practice satisfy this.

To enhance readability, operation  $+$  is called *additive operation*, while  $\times$  is called *multiplicative operation*. Note that we use a boldfaced  $+$  and symbol  $\times$  to avoid confusion with the additive and multiplicative operations over reals ( $+$  and  $\cdot$ ).

**Instances of c-semirings and application to QoS.** C-semirings are the formal structure of many QoS attributes. For example:

- $\langle \{true, false\}, \vee, \wedge, false, true \rangle$  (boolean): Network and service availability.
- $\langle \mathbb{R}^+, min, +, +\infty, 0 \rangle$  (optimization): Price, propagation delay.
- $\langle \mathbb{R}^+, max, min, 0, +\infty \rangle$  (max/min): Bandwidth.
- $\langle [0, 1], max, \cdot, 0, 1 \rangle$  (probabilistic): Performance and rates.
- $\langle [0, 1], max, min, 0, 1 \rangle$  (fuzzy): Performance and rates.
- $\langle 2^N, \cup, \cap, \emptyset, N \rangle$  (set-based, where  $N$  is a set): Capabilities and access rights.

**Example 1** Recall the example introduced in Section 1, where our systems have various agents and resources. Assume that the attributes associated to the usage of a resource by an agent are a certain price to pay and access rights, respectively represented by an optimization c-semiring  $C_{opt}$  and a set-based c-semiring  $C_{set} = \langle 2^N, \cup, \cap, \emptyset, N \rangle$ , with  $N = \{a_0, \dots, a_n\}$  being the set of access rights.

**C-Semirings and Lattices.** The additive operation of the c-semiring induces a partial order as follows:  $a \leq_S b$  iff  $a + b = b$ . For example, in the optimization c-semiring  $C_{opt}$ ,  $\leq_S$  corresponds to the arithmetic relation  $\geq$ , while in  $C_{set}$  it corresponds to set inclusion  $\subseteq$ .

One can show that  $\leq_S$  is indeed a partial order, that  $+$ ,  $\times$  are monotone over  $\leq_S$ ,  $\mathbf{0}$  and  $\mathbf{1}$  are respectively the minimum and maximum elements of  $\leq_S$ , and  $\langle A, \leq_S \rangle$  is a complete lattice [6]. Indeed for the examples  $C_{opt}$  and  $C_{set}$ ,  $\langle \mathbb{R}, \geq \rangle$  and  $\langle N, \subseteq \rangle$  are clearly complete lattices.

<sup>2</sup> When  $+$  is applied to a set with two elements we use  $+$  as binary operator in infix notation, while in all other cases we use symbol  $\sum$  in prefix notation.

In some instances of a c-semiring, the multiplicative operation is idempotent. This implies, among other things, that  $+$  distributes over  $\times$  and  $\langle A, \leq_S \rangle$  is a distributive lattice [6]. In such case we call the c-semiring *distributive c-semiring* or *distributive lattice*, but prioritize the former in order to avoid confusion between the algebraic structure and the implicit lattice since in some cases the implicit lattice is distributive but the c-semiring is not. C-semiring  $C_{set}$ , for instance, is a distributive c-semiring. On the other hand,  $C_{opt}$  is not a distributive c-semiring, since its multiplicative operation (real addition) is not idempotent. Note that, however, the partial order of  $C_{opt}$  induces a distributive lattice  $\langle \mathfrak{R}, \leq \rangle$ .

**Negation in C-Semirings.** C-Semirings have no negation operator in general. Consider, for instance, the classical De Morgan negation, i.e. a unary operator  $- : A \rightarrow A$ , such that  $-a \in A$  and  $--(a) = a$  for all  $a \in A$  (involution),  $-\sqcup\{A'\} = \sqcap\{-a \mid a \in A'\}$  for all  $A' \subseteq A$  (De Morgan) and  $a \leq b \Leftrightarrow -b \leq -a$  (antimonotonicity), where  $\sqcup$  and  $\sqcap$  are the *lowest upper bound* and *greatest lower bound* operators of the lattice  $\langle A, \leq_S \rangle$ . It is not possible to define a De Morgan negation for the optimization c-semiring  $C_{set}$ . Other c-semirings (fuzzy, boolean, max/min), however, can be equipped with a negation, resulting in most cases in boolean algebras. For example, in  $C_{set}$  set complement is a De Morgan negation. Note that the duality  $-(a+b) = (-a) \times (-b)$  holds exactly when  $\times$  is idempotent.

In order to tackle this problem we consider c-semirings to be equipped with a set  $F$  of functions, which range is  $A$  and which domain is  $A^i$  for  $i \geq 0$ . Set  $F$  is used to represent additional functions over the domain of the concrete c-semiring being considered other than the additive and the multiplicative operations. For example, if the c-semiring under consideration is the boolean one, then one can include boolean negation as a function of  $F$  in order to be able to define all possible boolean functions<sup>3</sup>. In our running examples, for instance, one might want to include arithmetic functions like real multiplication or division in  $C_{opt}$  or set complement in  $C_{set}$ . In the rest of the paper  $F$  refers to the set of all functions  $A^i$  for  $i \geq 0$  of the c-semiring under consideration.

**Composition of C-Semirings.** C-semiring based methods have a unique advantage when problems with multiple QoS criteria must be tackled: Cartesian products, exponentials and power constructions of c-semirings are c-semirings. Thus the same concepts and algorithms can be applied again and again.

The Cartesian product of  $n$  c-semirings  $C_i = \langle A_i, +_i, \times_i, \mathbf{0}_i, \mathbf{1}_i \rangle$  with  $i = 1..n$  is a c-semiring [6]  $C = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$  defined as follows:

- $A = A_1 \times \dots \times A_n$ ;
- $+$  is such that for any  $a, b \in A$ ,  $a+b = (a_1+_1b_1, \dots, a_n+_nb_n)$ ;

<sup>3</sup> Boolean negation, disjunction and conjunction are sufficient to define all boolean functions.

- $\times$  is such that for any  $a, b \in A$ ,  $a \times b = (a_1 \times_1 b_1, \dots, a_n \times_n b_n)$ ;
- $\mathbf{0} = (\mathbf{0}_1, \dots, \mathbf{0}_n)$ ;
- $\mathbf{1} = (\mathbf{1}_1, \dots, \mathbf{1}_n)$ .

The Hoare Power Domain of a c-semiring is of special interest since it let us formalize multi-criteria optimization problems. The Cartesian product of c-semirings is not suited to solve such problems since values of the original c-semirings are combined independently.

**Example 2** Let  $C = C_{opt} \times C_{set}$  and assume an agent can use a certain resource in two ways with QoS level  $(3, \{a_1\})$  or  $(2, \{a_1, a_2\})$ , respectively. Imagine we are interested in the best usage between both options. If we add both values we get  $(3, \{a_1\}) + (2, \{a_1, a_2\}) = (2, \{a_1, a_2\})$ , but observe that this value does not represent a valid value (it is not a choice between two values), but rather a lower bound. Clearly, both values are incomparable.

The solution to this problem is to use the Hoare power domain of the Cartesian product of the various optimization c-semirings as explained in [7].

Let  $C = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$  be a c-semiring. The Hoare power domain of  $C$ , denoted by  $P^H(C) = \langle P^H(A), \uplus, \times^*, \emptyset, A \rangle$  is a c-semiring [7] where:

- $P^H(A)$  is the Hoare Power Domain of  $A$ , i.e.,  $P^H(A) = \{S \subseteq A \mid x \in S, y \leq_S x \Rightarrow y \in S\}$ . In words,  $P^H(A)$  is the set of all subsets of  $A$  which are downward closed under the ordering  $\leq_S$ ;
- $\uplus$  is the formal union;
- $\times^*$  takes two sets and produces another set obtained by applying  $\times$  to each element of the first set with each element of the second one.

Intuitively, the Hoare Power Domain works with sets of non-dominated values.

**Example 3** Returning to our example, let  $C_{both} = P^H(C_{opt} \times C_{set})$ . Now, adding values  $(3, \{a_1\})$  and  $(2, \{a_1, a_2\})$  we get  $(3, \{a_1\}) + (2, \{a_1, a_2\})$  which is equal to  $\{(3, \{a_1\}), (2, \{a_1, a_2\})\}$ , i.e. we get a set containing the (incomparable) QoS levels of the two options to use the resource.

**Transition Systems.** As already advanced in the first section, we give an interpretation of our logics over transition systems. Our concept of transition system is the usual one extended with a c-semiring. More precisely, we call a transition system a tuple  $\langle S, T, C, P, I \rangle$ , where  $S$  is a set of states,  $T \subseteq S \times S$  is a set of transitions,  $C$  is a c-semiring,  $P$  is a set of propositions and  $I$  is a function mapping propositions of  $P$  into valuations, which are mappings  $S \rightarrow A$ , where  $A$  is the domain of the c-semiring under consideration. Abusing of notation interpretation also maps function names (drawn from a set  $\mathcal{F}$ ) into actual c-semiring functions. The type of  $I$  is thus  $(P \cup \mathcal{F}) \rightarrow ((S \rightarrow A) \cup F)$ .

We assume the transition system to be *image-finite*, i.e. that for any given  $s \in S$  we have that  $\{s' \mid (s, s') \in T\}$  is a finite set. We sometimes require  $T$  to be total, i.e. for every state  $s$  there is at least one outgoing transition  $(s, s') \in T$ . This avoids end states in the system. Runs of a system are *maximal paths* in the underlying state transition graph, i.e. paths that are either infinite or end in an end state. A path is a sequence  $s_0 s_1 s_2 \dots$ , such that for all  $i \geq 0$  we have  $s_i \in S$  and  $(s_i, s_{i+1}) \in T$ . We denote by  $|p|$  the length of a finite path  $p$ , by  $s_i^p$  the  $(i+1)$ -th state of path  $p$  and by  $\gamma(s)$  the set of runs starting at  $s$ .

We denote the concatenation of two paths  $p, q$  as  $pq$ , where we require  $p$  to be finite and  $(s_{|p|-1}^p, s_0^q) \in T$ , i.e. there is a transition from the last state of  $p$  to the first state of  $q$ . A *cycle* is a path  $p = qs$ , where  $s$  is equal to  $s_0^p$ , the initial state of  $p$ . Let us denote with  $p^i$  the cycle that results from repeating  $i$  times cycle  $p$ . More precisely,  $p^0 = s$  and  $p^{i+1} = qp^i$ .

In the following, let  $M = \langle S, T, C, P, I \rangle$  be the transition system under consideration.

### 3 Semiring Logics

In this section we first give the syntax and semantics of the  $\mu$ -calculus extension, which we call *c-semiring  $\mu$ -calculus*. Later we describe the syntax of the CTL extension, called *c-semiring CTL*, and give two semantics: the path and the fixpoint semantics, showing that the equivalence between the two semantics does not hold in general.

We interpret a formula of the extended logics as a mapping  $v : S \rightarrow A$ , i.e. a mapping from the set of system states  $S$  to the domain  $A$  of the c-semiring. We call such a mapping, a *valuation*. In the following let  $V = A^S$  be the set of all valuations. It is easy to see that the set of all valuations together the extension of  $+$  and  $\times$  to valuations forms a c-semiring. More precisely, let us define an additive operation  $\oplus : V \times V \rightarrow V$  such that  $(v \oplus v')(s) = v(s) + v'(s)$  for all  $s \in S$ . Similarly, we define a multiplicative operation  $\otimes : V \times V \rightarrow V$  as follows:  $(v \otimes v')(s) = v(s) \times v'(s)$  for all  $s \in S$ . Let  $\llbracket \mathbf{0} \rrbracket$  and  $\llbracket \mathbf{1} \rrbracket$  be the valuations that respectively assign  $\mathbf{0}$  and  $\mathbf{1}$  to every state. It can be easily shown that  $\langle V, \oplus, \otimes, \llbracket \mathbf{0} \rrbracket, \llbracket \mathbf{1} \rrbracket \rangle$  is a c-semiring, since it can be seen as  $C^{|S|}$  and we know that the Cartesian product of two c-semirings is a c-semiring [6]. The partial order of this c-semiring is denoted by  $\leq_V$ . It is easy to see that for any two valuations  $v, v' \in V$ ,  $v \leq_V v'$  iff for all  $s \in S$  we have  $v(s) \leq_S v'(s)$ . Then  $\langle V, \leq_V \rangle$  is a complete lattice. The defined c-semiring inherits various of the properties of  $C$ . For example, if  $\times$  is idempotent, so is  $\otimes$ . Hence, in the document we sometimes refer to original properties of  $+$  and  $\times$  instead of referring to properties of  $\otimes$  and  $\oplus$ .

Let  $Z$  be a set of valuation variables and  $AP \subseteq P$  be a set of *atomic propositions*. Valuation variables are used inside fixpoint formulae as explained below, while set  $AP$  generalizes the set of atomic propositions of boolean logics. Indeed, an atomic boolean proposition  $p$  can be interpreted as a valuation that assigns *true* to states where  $p$  holds and *false* otherwise. Let  $p \in AP$ ,  $z \in Z$  and  $f \in \mathcal{F}$ . Formulae of  $c\text{-}L_\mu$  are defined as follows:

$$\begin{aligned} \phi &::= \mathbf{0} \mid \mathbf{1} \mid p \mid z \mid f(\phi, \dots, \phi) \mid \phi + \phi \mid \phi \times \phi \mid \kappa \mathbf{X}\phi \mid \mu z. \phi \mid \nu z. \phi \\ \kappa &::= \sqcap \mid \sqcup \mid \Pi \end{aligned}$$

Note that constants other than  $\mathbf{0}$  and  $\mathbf{1}$  are not introduced explicitly but as functions. Operators  $\mu$  and  $\nu$  are used to express least and greatest fixpoints, respectively. Temporal operator  $\mathbf{X}$  is used to refer to the evaluation of a formula in a next state. Since a state might have more than one immediate successor we use a quantifier  $\kappa$  preceding  $\mathbf{X}$ .

While boolean  $\mu$ -calculus considers only two quantifiers, namely existential and universal quantification, we consider three possible cases:  $\sqcap$ ,  $\sqcup$  and  $\Pi$ . As we shall see they respectively express the application of the greatest lower bound, additive and multiplicative operations of the lattice  $\langle A, \leq_S \rangle$ . Recall that in  $c$ -semirings,  $\sqcup$  coincides with  $\sum$  [6]. For example, in  $C_{opt}$  both the additive operation and the lowest upper bound are *min*, while in  $C_{set}$  both are  $\cup$ . Hence, we do not include  $\sqcup$  as a quantifier, since it is redundant. Note that  $\sqcap$  is not introduced explicitly in the syntax, but it can be used as a function in  $F$ . Finally, recall that  $\sqcap$  coincides with  $\times$  if  $\times$  is idempotent [6]. This is indeed the case in  $C_{set}$  since both operations are set intersection  $\cap$ , but not in  $C_{opt}$  where greatest lower bound is *max* while the multiplicative operation is real addition  $+$ .

**Example 4** Recall our running example and the property mentioned in Section 1: *which is the best QoS level (of the usage of the resource by the agent) ever achieved?*. Assume that the QoS level of the usage is given by the atomic proposition  $p$ . Then, the formula that expresses the property is  $\mu z. p + \sum \mathbf{X}z$ . Roughly speaking the intuition behind the formula is that the best QoS for the usage is the one of the current state or the best amongst the QoS ever achieved from the next states.

**Semantics.** We interpret a formula  $\phi$  as a valuation  $\llbracket \phi \rrbracket_e$ , where  $e : Z \rightarrow V$  is an environment:

$$\begin{aligned}
\llbracket \mathbf{0} \rrbracket_e(s) &= \mathbf{0} & \llbracket \phi_1 \times \phi_2 \rrbracket_e &= \llbracket \phi_1 \rrbracket_e \otimes \llbracket \phi_2 \rrbracket_e \\
\llbracket \mathbf{1} \rrbracket_e(s) &= \mathbf{1} & \llbracket \kappa \mathbf{X} \phi \rrbracket_e(s) &= \kappa_{(s,s') \in T} \llbracket \phi \rrbracket_e(s') \\
\llbracket p \rrbracket_e &= I(p) & \llbracket \nu z. \phi \rrbracket_e &= \mathit{FIX} \lambda z'. \llbracket \phi \rrbracket_{e[z'/z]} \\
\llbracket z \rrbracket_e &= e(z) & \llbracket \mu z. \phi \rrbracket_e &= \mathit{fix} \lambda z'. \llbracket \phi \rrbracket_{e[z'/z]} \\
\llbracket \phi_1 + \phi_2 \rrbracket_e &= \llbracket \phi_1 \rrbracket_e \oplus \llbracket \phi_2 \rrbracket_e & \llbracket f(\phi_1, \dots, \phi_n) \rrbracket_e(s) &= I(f)(\llbracket \phi_1 \rrbracket_e(s), \dots, \llbracket \phi_n \rrbracket_e(s))
\end{aligned}$$

where  $s \in S$ ,  $\kappa \in \{\sqcap, \sqcup, \sqcap\}$ ,  $\mathit{FIX}$  abbreviates *greatest fixpoint* and  $\mathit{fix}$ , *least fixpoint*, and  $e[v/z]$  is the same as  $e$  except that  $e[v/z](z) = v$ . A formula  $\phi$  is closed if every variable is bound by a fixpoint operator. In such cases  $\llbracket \phi \rrbracket_e$  does not depend on  $e$  and we just write  $\llbracket \phi \rrbracket$ . Functions  $\lambda z'. \llbracket \phi \rrbracket_{e[z'/z]}$  are operators on  $V$  (functions  $V \rightarrow V$ ) that we will denote with  $\tau$ .

In the rest of the paper we require recursion variables to appear as operands of monotone functions only. With this restriction, it is easy to see that every possible operator  $\tau$  is monotone. Hence, by Knaster-Tarski Theorem [27] the fixpoints are well-defined. More precisely,  $\mu z. \tau(z) = \sqcap_V \{v \mid v \leq_V \tau(v)\}$  (equivalently  $\sqcap_V \{v \mid v = \tau(v)\}$ ) and  $\nu z. \tau(z) = \sqcup_V \{v \mid v \leq_V \tau(v)\}$  (equivalently  $\sqcup_V \{v \mid v = \tau(v)\}$ )<sup>4</sup>.

In addition, if  $\tau$  is  $\sqcup_V$ -continuous, then  $\mu z. \tau(z) = \sqcup_V \{\tau^i(\mathbf{0})\}$  and if it is  $\sqcap_V$ -continuous, then  $\nu z. \tau(z) = \sqcap_V \{\tau^i(\mathbf{1})\}$ , where  $\tau^0(z) = z$  and  $\tau^{i+1}(z) = \tau(\tau^i(z))$  for  $i = 1, 2, \dots$ . In the rest of the paper we consider continuous operators. Note that, because  $\tau$  is monotone,  $\sqcup_V \{\tau^i(\mathbf{0})\} = \tau^\infty(\mathbf{0})$  and  $\sqcap_V \{\tau^i(\mathbf{1})\} = \tau^\infty(\mathbf{1})$ . Hence, fixpoint iteration can be applied to evaluate a formula. Nevertheless this method is not always feasible as we shall see in a next section.

**C- $L_\mu$  with negation.** The existence of a negation operator as defined in Section 2, enables some equivalences between formulae, for instance  $\neg(\sum \mathbf{X} \phi) = \sqcap \mathbf{X} \neg \phi$  and  $\neg(\mu z. \phi) = \nu z. \neg(\phi[\neg z/z])$ . Note that  $\neg(\sum \mathbf{X} \phi)$  is equivalent to  $\sqcap \mathbf{X} \neg \phi$  if  $\times$  is idempotent.

As in boolean  $\mu$ -calculus, one has to impose syntax restrictions on the use of negations in order to guarantee monotony. More precisely, we shall require that each variable appears under an even number of negations, and for each function  $f \in F$  there is a (dual) function  $\bar{f} \in F$  such that  $\neg f(\phi_1, \dots, \phi_n) = \bar{f}(\neg \phi_1, \dots, \neg \phi_n)$ . We have also to require the existence of a dual for  $\otimes$ . The idea is that with these dualities and the syntactic restriction one can push negations such that they appear applied to atomic propositions only.

<sup>4</sup> In this case,  $\sqcup_V$  and  $\sqcap_V$  respectively refer to the *lowest upper bound* and *greatest lower bound* operations of the lattice  $\langle V, \leq_V \rangle$ .

**Syntax.** The syntax of c-CTL is defined as follows:

$$\begin{aligned}
 \phi &::= \mathbf{0} \mid \mathbf{1} \mid z \mid p \mid f(\phi, \dots, \phi) \mid \phi + \phi \mid \phi \times \phi \mid \kappa\psi \mid \kappa\mathbf{X}\phi \\
 \kappa &::= \prod \mid \sum \mid \Pi \\
 \psi &::= [\phi \mathbf{T}_{\odot}^{\circ} \phi] \\
 \odot, \otimes &::= + \mid \times
 \end{aligned}$$

C-CTL formulae are state formulae generated by  $\phi$ . Symbol  $\kappa$  is used to introduce quantifiers over path formulae, which are generated by  $\psi$ . Path formulae combine state formulae with temporal operators  $\mathbf{X}$  (next time), and  $\mathbf{T}_{\odot}^{\circ}$ . As we shall see the first refers to the evaluation of a formula in a next state, while the second uses two operators  $\odot$  and  $\otimes$ : one to quantify over prefixes of a path and the other to quantify over states of a prefix. As syntactic restriction we require  $\odot$  and  $\otimes$  to be different.

We shall sometimes use some abbreviations, for instance,  $\mathbf{U}$  for  $\mathbf{T}_{+}^{\times}$ ,  $\mathbf{R}$  for  $\mathbf{T}_{\times}^{+}$ ,  $\mathbf{F}\phi$  for  $[1\mathbf{T}_{\times}^{+}\phi]$  and  $\mathbf{G}\phi$  for  $[0\mathbf{T}_{+}^{\times}\phi]$ . Not surprisingly,  $\mathbf{U}$ ,  $\mathbf{R}$ ,  $\mathbf{F}$  and  $\mathbf{G}$  correspond, in a boolean setting, to the CTL temporal operators *until*, *release*, *eventually* and *globally*.

Indeed the main motivations behind the syntax of c-CTL was to extend CTL to c-semirings, by substituting existential quantification and boolean disjunction by the additive operation, and universal quantification and boolean conjunction by the multiplicative operation. Note that the syntactic restrictions imposed, which semantic consequences are discussed after the presentation of the semantics, are enough to cover CTL.

**Semantics.** Again, we interpret a state formula as a mapping from the set of states  $S$  to the set of c-semiring values  $A$ . Path formulae, instead, assign c-semiring values to paths. The semantics of  $\mathbf{0}$ ,  $\mathbf{1}$ ,  $p$ ,  $z$ ,  $f(\phi_1, \dots, \phi_n)$ ,  $\phi_1 + \phi_2$  and  $\phi_1 \times \phi_2$  is the same as in  $c\text{-}L_{\mu}$ . The rest is defined as follows:

$$\begin{aligned}
 \llbracket \kappa\mathbf{X}\phi \rrbracket(s) &= \kappa_{(s,s') \in T} \llbracket \phi \rrbracket(s') \\
 \llbracket \kappa\psi \rrbracket(s) &= \kappa_{p \in \gamma(s)} \llbracket \psi \rrbracket(p) \\
 \llbracket \phi_1 \mathbf{T}_{\odot}^{\circ} \phi_2 \rrbracket(p) &= \odot_{i \geq 0} (\llbracket \phi_2 \rrbracket(s_i^p) \otimes \otimes_{0 \leq j < i} \llbracket \phi_1 \rrbracket(s_j^p)),
 \end{aligned}$$

where a  $p$  is a path. In words, the semantics of  $\phi_1 \mathbf{T}_{\odot}^{\circ} \phi_2$  means that for every prefix of  $p$ ,  $\otimes$  is applied to the evaluation of  $\phi_1$  in every state of the prefix but the last one and  $\phi_2$  in that state. The values for every prefix are quantified using  $\odot$ .

It can be shown that requiring  $\otimes$  to be different from  $\odot$  does not impede to express  $[\phi_1 \mathbf{T}_+^+ \phi_2]$  since it is equivalent to  $\mathbf{F}(\phi_1 + \phi_2)$ . To the contrary, there is no way to express  $\phi_1 \mathbf{T}_\times^+ \phi_2$ , because  $\times$  is not idempotent in general. However, we are not sure about the applicability of such operator. Note that we could have included  $\sqcap$  as one of  $\odot, \otimes$  increasing the expressivity of the logic, but, for the sake of simplicity, we have decided to let such as an extension as a further research topic.

**Example 5** We are now ready to formulate property *which is the best QoS level (of the usage of the resource by the agent) ever achieved?* of our running example in a more intuitive way. Namely, the corresponding formula is  $\Sigma \mathbf{F}p$ , where  $p$  represents the QoS level of the usage of the resource. The intuition is clear: for each path starting at the state where we evaluate the formula we select by c-semiring addition the best QoS level, and amongst all paths we select the best such value again.

The previous semantics is called *path semantics*. In model checking, there is an alternative equivalent semantics which interprets CTL formulae as fixpoints [10]. For example, CTL formula  $\mathbf{EF}\phi$  is interpreted as  $\mu z. \phi \vee \mathbf{E}Xz$ . This equivalence shows that boolean  $\mu$ -calculus contains CTL, and allows to perform CTL model checking formulae by fixpoint iteration. This suggests to define the following fixpoint semantics to c-CTL formulae:

$$\begin{aligned} \llbracket \kappa[\phi_1 \mathbf{T}_\times^+ \phi_2] \rrbracket^f &= \llbracket \mu z. \phi_2 + (\phi_1 \times \kappa \mathbf{X}z) \rrbracket \\ \llbracket \kappa[\phi_1 \mathbf{T}_+^+ \phi_2] \rrbracket^f &= \llbracket \nu z. \phi_2 \times (\phi_1 + \kappa \mathbf{X}z) \rrbracket, \end{aligned}$$

where for any other formula  $\phi$  the fixpoint semantics is exactly the path semantics. We will see that the path and fixpoints semantics of c-CTL are not equivalent in general. The main reason is that the additive operation does not distribute over the multiplicative one, in general.

**Example 6** Consider our running example and assume that the only QoS attribute considered is the price to pay for the usage, represented by c-semiring  $C_{opt}$ . Clearly,  $\min$  does not distribute over  $+$ . Suppose we have a transition system with three states  $s_0, s_1$  and  $s_2$ , where neither of  $s_1, s_2$  has a transition and  $s_0$  has two transitions: one to  $s_1$  and one to  $s_2$ . Let  $p$  represent the price to pay for the usage of the resource, such that the price is constant 1. It is easy to see that  $\llbracket \Pi \mathbf{F}p \rrbracket(s_0) = \min(1, 1) + \min(1, 1) = 2$ , while  $\llbracket \mu z. \min(p, \Pi \mathbf{X}z) \rrbracket(s_0) = \min(1, 1 + 1) = 1$ . The semantics of both formulae differ even if we introduce self-transitions to  $s_1$  and  $s_2$  to avoid end states.

What is the meaning of the two semantics of the formulae? Suppose that in the scenario agents can clone themselves. Consider two possibilities for a cloning agent. The first one is that when an agent finds a branching in its behavior then it makes one copy to explore each of the branches. The second strategy

consists in making one clone for each possible behaviour in advance. The path semantics of formula  $\mathbf{F}$  represents the multiplication for each clone of the best QoS ever achieved. The fixpoint semantics instead, represent the best QoS states where the cloning and exploration process should stop.

In order show the relation between both semantics we need to define a bounded version of the temporal operator  $\mathbf{T}$ . Intuitively, the idea is that they take into account the first  $k$  states of a path only:

$$\llbracket [\phi_1 \mathbf{T}_{\odot}^{\odot, k} \phi_2] \rrbracket(p) = \odot_{0 \leq i < k} (\phi_2(s_i^p) \odot \odot_{0 \leq j < i} \phi_1(s_j^p))$$

Clearly,  $\mathbf{T}_{\odot}^{\odot, \infty} \equiv \mathbf{T}_{\odot}^{\odot}$ . We use a theorem to state under which conditions are the path and fixpoint semantics equivalent.

**Theorem 7** *If  $T$  is total,  $\kappa$  is idempotent,  $\odot$  distributes over  $\kappa$  and  $\odot$ , and  $\odot$  distributes over  $\kappa$  then  $\llbracket \kappa[\phi_1 \mathbf{T}_{\odot}^{\odot} \phi_2] \rrbracket = \llbracket \kappa[\phi_1 \mathbf{T}_{\odot}^{\odot, k} \phi_2] \rrbracket^f$ .*

**PROOF.** Let  $\tau$  be  $\lambda z. \llbracket \phi_2 \odot (\phi_1 \odot \kappa \mathbf{X} z) \rrbracket$ . We use  $\mathbf{T}^k$ , the bounded version of  $\mathbf{T}$ , and induction on  $k$ . We will show the theorem for  $\odot = +$  and  $\odot = \times$ . The proof for the other just requires to replace  $\mathbf{0}$  by  $\mathbf{1}$ .

For  $k = 1$  it is easy to see that  $\llbracket \kappa[\phi_1 \mathbf{T}_{\odot}^{\odot, k} \phi_2] \rrbracket = \tau^k(\mathbf{0})$  by definition of  $\llbracket \cdot \rrbracket^f$  since  $\odot = +$  and  $\odot = \times$ . More precisely,  $\llbracket \kappa[\phi_1 \mathbf{T}_{\odot}^{\odot, 1} \phi_2] \rrbracket(s)$  is  $\kappa_{p \in \gamma(s)} \llbracket \phi_2 \rrbracket(s)$ , while  $\tau(\mathbf{0})$  is  $\llbracket \phi_2 \rrbracket(s)$ . Since  $\kappa$  is idempotent, the theorem holds for  $k = 1$ .

Assume for induction that  $\llbracket \kappa[\phi_1 \mathbf{T}_{\odot}^{\odot, n} \phi_2] \rrbracket = \tau^n(\mathbf{0})$ , then applying  $\tau$  to both sides of the equality we get  $\tau(\llbracket \kappa[\phi_1 \mathbf{T}_{\odot}^{\odot, n} \phi_2] \rrbracket) = \tau^{n+1}(\mathbf{0})$ . But, by definition of  $\tau$ ,  $\tau(\llbracket \kappa[\phi_1 \mathbf{T}_{\odot}^{\odot, n} \phi_2] \rrbracket) =$

$$\begin{aligned} & \llbracket \phi_2 \odot (\phi_1 \odot \kappa \mathbf{X} \llbracket \kappa[\phi_1 \mathbf{T}_{\odot}^{\odot, n} \phi_2] \rrbracket) \rrbracket(s) = \\ & \hspace{10em} (\text{Various def.}) \\ & \llbracket \phi_2 \rrbracket(s) \odot (\llbracket \phi_1 \rrbracket(s) \odot \kappa_{p \in \gamma(s)} \odot_{1 \leq i < n} (\llbracket \phi_2(s_i^p) \rrbracket \odot \odot_{1 \leq j < i} \llbracket \phi_1(s_j^p) \rrbracket)) = \\ & \hspace{10em} (\odot \text{ distr. over } \odot) \\ & \llbracket \phi_2 \rrbracket(s) \odot \kappa_{p \in \gamma(s)} (\llbracket \phi_1 \rrbracket(s) \odot \odot_{1 \leq i < n} (\llbracket \phi_2(s_i^p) \rrbracket \odot \odot_{1 \leq j < i} \llbracket \phi_1(s_j^p) \rrbracket)) = \\ & \hspace{10em} (\odot \text{ distr. over } \odot) \\ & \llbracket \phi_2 \rrbracket(s) \odot \kappa_{p \in \gamma(s)} (\odot_{1 \leq i < n+1} (\llbracket \phi_1 \rrbracket(s) \odot \llbracket \phi_2(s_i^p) \rrbracket \odot \odot_{1 \leq j < i} \llbracket \phi_1(s_j^p) \rrbracket)) = \\ & \hspace{10em} (\odot \text{ distr. over } \odot) \\ & \llbracket \phi_2 \rrbracket(s) \odot \kappa_{p \in \gamma(s)} (\sum_{1 \leq i < n+1} (\llbracket \phi_2(s_i^p) \rrbracket \odot \odot_{0 \leq j < i} \llbracket \phi_1(s_j^p) \rrbracket)) = \\ & \hspace{10em} (\odot \text{ distr. over } \kappa) \end{aligned}$$

$$\begin{aligned}
\kappa_{p \in \gamma(s)}(\llbracket \phi_2 \rrbracket(s) \odot \sum_{1 \leq i < n+1} (\llbracket \phi_2(s_i^p) \rrbracket \otimes \otimes_{0 \leq j < i} \llbracket \phi_1 \rrbracket(s_j^p))) &= \\
& (\otimes_{0 \leq j < 0} \llbracket \phi_1 \rrbracket(s_j^p) \text{ is the unit of } \otimes) \\
\kappa_{p \in \gamma(s)}(\llbracket \phi_2 \rrbracket(s) \odot \sum_{0 \leq i < n+1} (\llbracket \phi_2(s_i^p) \rrbracket \otimes \otimes_{0 \leq j < i} \llbracket \phi_1 \rrbracket(s_j^p))) &= \\
& (\text{Various def.}) \llbracket \kappa[\phi_1 \mathbf{T}_{\otimes}^{\odot, n+1} \phi_2] \rrbracket
\end{aligned}$$

□

Note that by definition,  $+$  and  $\sqcap$  are idempotent and both  $+$  and  $\times$  distribute over  $+$ , but the rest of the distributions and idempotency of  $\times$  are not true in general. However, if  $\times$  is idempotent, then  $\times$  coincides with  $\sqcap$  and both  $+$  and  $\times$  distribute over  $\times$ . Hence, if  $T$  is total and the c-semiring under consideration is distributive the path and fixpoint semantics are equivalent. The relation between the path and fixpoint semantics can be refined.

**Theorem 8** *The following inequalities hold:*

$$\begin{aligned}
\llbracket \Sigma[\phi_1 \mathbf{T}_{\times}^+ \phi_2] \rrbracket &= \llbracket \Sigma[\phi_1 \mathbf{T}_{\times}^+ \phi_2] \rrbracket^f & \llbracket \Sigma[\phi_1 \mathbf{T}_{+}^{\times} \phi_2] \rrbracket &\geq_S \llbracket \Sigma[\phi_1 \mathbf{T}_{\times}^+ \phi_2] \rrbracket^f \\
\llbracket \sqcap[\phi_1 \mathbf{T}_{\otimes}^{\circ} \phi_2] \rrbracket &\geq_S \llbracket \sqcap[\phi_1 \mathbf{T}_{\otimes}^{\circ} \phi_2] \rrbracket^f & \llbracket \sqcap[\phi_1 \mathbf{T}_{\otimes}^{\circ} \phi_2] \rrbracket &\leq_S \llbracket \sqcap[\phi_1 \mathbf{T}_{\otimes}^{\circ} \phi_2] \rrbracket^f
\end{aligned}$$

**PROOF.** The proof is similar to that of Theorem 7, where one substitutes equalities with the corresponding inequalities when applying distribution, namely  $a+(b+c) = (a+b)+(a+c)$ ,  $a \times (b+c) = (a \times b)+(a \times c)$ ,  $a+(b \times c) \geq_S (a+b) \times (a+c)$ ,  $a \times (b \times c) \geq_S (a \times b) \times (a \times c)$ ,  $a+(b \sqcap c) \leq_S (a+b) \sqcap (a+c)$  and  $a \times (b \sqcap c) \leq_S (a \times b) \sqcap (a \times c)$ , which are fairly easy to show. □

As a consequence it is easy to see that for any formula  $\phi$  of the  $\sqcap$ -free fragment of c-CTL, c- $\sqcap$ CTL for short, i.e. the fragment of c-CTL where  $\sqcap$  is not used as quantifier, we have  $\llbracket \phi \rrbracket \geq_S \llbracket \phi \rrbracket^f$ . **C-CTL with Negation.** As in  $c-L_{\mu}$ , we can incorporate a negation operator. If we want to have the usual equivalence  $-\Sigma[\phi_1 \mathbf{U} \phi_2] \equiv \sqcap[-\phi_1 \mathbf{R} -\phi_2]$ , we need  $\times$  to be idempotent.

#### 4 On computing the semantics.

Now we come to the problem of evaluating formulae. First of all, using fixpoint iteration to compute  $c-L_{\mu}$  is not always possible; continuous operators are required. Even then, fixpoint iteration becomes infeasible for some formulae if the transition system or the c-semiring domain are infinite.

**Example 9** Recall our running example. Consider the evaluation of  $\nu z.p \times \Sigma \mathbf{X}z$  (or equivalently  $\Sigma \mathbf{G}p$ ) representing the best QoS of the cumulative

usage of the resource in infinite executions. Assume the transition system has just one state  $s$  and a transition  $(s, s)$ , where the  $c$ -semiring is  $C_{opt}$  and  $p$  is the price to pay for the resource, which evaluated to a positive value  $a > 0$  in state  $s$ . The value of the formula in  $s$  is clearly  $\infty$  but its computation requires infinitely many iterations.

While restricting to finite transition systems is reasonable, we cannot neglect infinite  $c$ -semiring domains. However, the fact that the path and fixpoints semantics of  $c$ -CTL are not equivalent in general, avoids the usual bypass of CTL algorithms as fixpoint iterations. Hence,  $c$ -CTL requires specific algorithms.

We first concentrate on distributive  $c$ -semirings. We claim that the model checking problem is decidable for formulae in  $c$ - $L_\mu$  and give an upper bound on the complexity. Next we refine the bound for  $c$ -CTL formulae, showing that the number of fixpoint iterations needed is proportional to the number of states of the system. Last we show how to evaluate formulae of some fragments of  $c$ -CTL when the  $c$ -semiring is not distributive.

#### 4.1 Model checking $c$ - $L_\mu$ for distributive $c$ -semirings.

We claim that fixpoint iteration (like in algorithm *eval* sketched below) is feasible when computing formulae of a fragment of  $c$ - $L_\mu$ . The fragment under consideration restricts to formulae in which recursion variables appear as operands of  $c$ -semiring operators  $+$ ,  $\times$  and temporal operator  $\mathbf{X}$  only. In other words, we forbid functions in  $F$  to be applied to recursion variables. We denote this fragment as  $c$ - $L_\mu^-$ .

The basic idea is that even if the domain of the  $c$ -semiring, and hence the corresponding lattice, are infinite, only a finite sublattice is involved. The reason lies in the fact that the sublattice generated by a finite set of elements of a (possibly infinite) distributive lattice is finite. We give some definitions before stating the theorem.

Let  $m(\phi)$  denote the set of all maximal subformulae of  $\phi$  that do not contain variables. Given a set  $M$  of formulae we denote by  $g(M)$  the set of values to which each formulae evaluates in each state of  $S$ . More formally,  $g(M) = \{\llbracket \phi \rrbracket(s) \mid \phi \in M, s \in S\}$ . In addition let  $FD(A')$  denote the (domain of the) free distributive lattice generated by a finite subset of  $A$ . Determining the size of a free distributive lattice is an open problem, but some asymptotic results exist [20]. A coarse bound on the size of  $FD(A)$  is  $2^{(2^A)}$ . In particular cases, like in the examples presented, this bound can be refined. For instance, in the set-based  $c$ -semiring  $C_{set}$  of our running example the size of  $FD(N)$  is bounded by  $2^N$  and if the order of the lattice is linear, as in the fuzzy and max/min  $c$ -semirings, the  $|FD(A)|$  is just  $|A|$ . Finally,  $h(A)$  denotes the height of a finite lattice (with domain)  $A$ . Unfortunately, we are not aware of results

about the height of free distributive lattice generated by a finite set. A coarse upper bound is of course the size of the lattice.

**Theorem 10** *If  $C$  is distributive,  $S$  finite and  $T$  total, any fixpoint formula  $\mu.\phi \in c-L_\mu^-$  can be computed using  $O(h(FD(g(\phi)))^{|S|})$  iterations.*

**PROOF.** A valuation  $v$  can be seen as a vector  $(x_0, x_1, \dots, x_n)$ , where  $S = \{s_0, s_1, \dots, s_n\}$  and  $x_i = v(s_i)$ . Least fixpoint iteration can be seen as an ascending sequence of valuations  $\tau^0 \leq_S \tau^1 \leq_S \dots$ , where  $\tau^{i+1} = \tau(\tau^i)$  for  $i \geq 0$  and  $\tau^0$  being the valuation that maps every state to  $\mathbf{0}$ . Valuation transformer  $\tau$  is just the function  $\lambda z'.\phi\{z'/z\}$ . Let us denote with  $x_i^j$  the value assigned to state  $s_i$  in the  $j$ -th iteration and let  $\tau_j$  denote  $\tau$  applied to  $s_j$ . Then we can write:  $x_i^{j+1} = \tau_i^j(x_0^j, x_1^j, \dots, x_n^j)$  and  $x_i^0 = \mathbf{0}$ .

Each  $x_i^{j+1}$  necessarily belongs to  $FD(g(\phi) \cup \{\mathbf{0}\})$ , since  $\tau_i^j$  only involves the lattice operations and values  $\mathbf{0}$ ,  $x_0^j, x_1^j, \dots, x_n^j$  and those in  $g(\phi)$ . This can be shown by induction on  $j$  in a fairly easy way. Therefore, the number of (distinct) valuations in the chain is clearly bounded by  $m^{|S|}$ , where  $m = h(FD(g(\phi) \cup \{\mathbf{0}\}))$ , because the chain of valuations is monotone.  $\square$

---

**Algorithm** *eval*( $M, \phi, e$ )

**Input:** A transition system  $M$ , where  $C$  is distributive, a  $c-L_\mu$  formula  $\phi$  and an environment  $e$ .

**Output:** A valuation as an array of  $|S|$  elements.

**switch**  $\phi$  **do**

**case**  $\mathbf{0}$  **return**  $(\mathbf{0}..\mathbf{0})$ ;

**case**  $\mathbf{1}$  **return**  $(\mathbf{1}..\mathbf{1})$ ;

**case**  $p$  **return**  $(I(p)[s_1]..I(p)[s_{|S|}])$ ;

**case**  $\phi_1 \mp \phi_2$  **return**

$(\text{eval}(M, \phi_1, e)[s_1] \mp \text{eval}(M, \phi_1, e)[s_1], \dots, \text{eval}(M, \phi_1, e)[s_{|S|}] \mp \text{eval}(M, \phi_1, e)[s_{|S|}])$ ;

**case**  $\phi_1 \mp \phi_2, f(\phi_1, \dots, \phi_n)$

    /\* similar as above \*/

**case**  $\kappa X\phi_1$

$v' := \text{eval}(M, \phi_1, e)$ ;

**foreach**  $s \in S$  **do**  $v[s] := \kappa_{(s,s') \in T} v'[s']$ ;

**return**  $v$  ;

**case**  $\mu z.\phi_1$

$v := (\mathbf{0}..\mathbf{0})$ ;

**repeat**  $v' := v; v := \text{eval}(M, \phi_1, e[v'/z])$  **until**  $v' = v$ ;

**return**  $v$ ;

**case**  $\nu z.\phi_1$

    /\* similar as above \*/

**end**

---

## 4.2 Model checking c-CTL for distributive c-semirings.

Concentrating on c-CTL one can show that only  $|S|$  iterations are necessary to evaluate a formula in the theorem below. For the boolean case the result was already known [15]. Now, we define some auxiliary results. We abbreviate  $(\prod_{0 \leq i < j} \llbracket \phi_1 \rrbracket(s_i^p)) \times \llbracket \phi_2 \rrbracket(s_j^p)$  with  $t_p(s_0^p \dots s_j^p)$  in the following. We call it a *term*. Observe that  $\llbracket \phi_1 \mathbf{U}^k \phi_2 \rrbracket$  is  $\prod_{0 \leq j < k} t_p(s_0^p \dots s_j^p)$ . We first define some lemmas.

**Lemma 11** *Let  $p$  be an infinite path such that  $s_i^p = s_j^p$  for some  $0 \leq j < i$ . Then  $\llbracket \phi_1 \mathbf{U}^{i+1} \phi_2 \rrbracket(p) = \llbracket \phi_1 \mathbf{U}^i \phi_2 \rrbracket(p)$ .*

**PROOF.** Observe that  $\llbracket \phi_1 \mathbf{U}^{i+1} \phi_2 \rrbracket(p) = \llbracket \phi_1 \mathbf{U}^i \phi_2 \rrbracket(p) + t_p(s_0^p \dots s_i^p)$ . Because  $s_i^p$  is exactly one  $s_j^p$  with  $j < i$  we have a term  $t_p(s_0^p \dots s_j^p)$  in  $\llbracket \phi_1 \mathbf{U}^i \phi_2 \rrbracket(p)$ . By absorption  $t_p(s_0^p \dots s_j^p) + t_p(s_0^p \dots s_i^p)$  is  $t_p(s_0^p \dots s_j^p)$ . Hence,  $\llbracket \phi_1 \mathbf{U}^{i+1} \phi_2 \rrbracket(p) = \llbracket \phi_1 \mathbf{U}^i \phi_2 \rrbracket(p)$ .  $\square$

**Lemma 12** *Let  $p = p'c^\omega$  be a path such that  $c = qs$  is a cycle. Then  $\llbracket \phi_1 \mathbf{U}^k \phi_2 \rrbracket(p) = \llbracket \phi_1 \mathbf{U}^{|p'q|} \phi_2 \rrbracket(p)$  for every  $k \geq |p'q|$ .*

**PROOF.** See that Lemma 11 holds for every  $k \geq |pq|$ .  $\square$

**Lemma 13** *Let  $p$  be an infinite path such that  $s_i^p = s_j^p$  for some  $0 \leq i < j$ . Let  $p'$  be a path  $s_0^p \dots s_{i-1}^p c^\omega$ , where  $c = s_i^p \dots s_j^p$ . For every prefix  $p'_1$  of  $p'$  with  $|p'_1| < j$ , there is a prefix  $p_1$  of  $p$  such that  $t_p(p'_1) = t_p(p_1)$ .*

**PROOF.** For every prefix of  $p'$  of length less than  $j$  there is an identical prefix of  $p$  since, up to the  $j$ -th state both paths are equal.  $\square$

**Theorem 14** *If  $C$  is distributive,  $S$  finite and  $T$  total then  $\llbracket \kappa[\phi_1 \mathbf{T}_\circ^{\circledast} \phi_2] \rrbracket = \llbracket \kappa[\phi_1 \mathbf{T}_\circ^{\circledast, |S|} \phi_2] \rrbracket$*

**PROOF.** We give the proof for  $\llbracket \phi_1 \mathbf{U} \phi_2 \rrbracket$ ; the rest of the cases are very similar. We proof by induction that  $\llbracket \prod[\phi_1 \mathbf{U}^k \phi_2] \rrbracket = \llbracket \prod[\phi_1 \mathbf{U}^{|S|} \phi_2] \rrbracket$  for  $k \geq |S|$ . This holds trivially for  $k = |S|$ . Assume for induction that it holds for  $k = n \geq |S|$ . We shall see that it holds for  $k = n + 1$ .

Consider a path  $p \in \gamma(s)$  and its  $n + 1$ -st state. There are two cases: if it coincides with a state preceding it in  $p$  we know that  $\llbracket \phi_1 \mathbf{U}^{n+1} \phi_2 \rrbracket(p) = \llbracket \phi_1 \mathbf{U}^n \phi_2 \rrbracket(p)$  by Lemma 11.

The second case is when the  $n + 1$ -st state does not coincide with any preceding state. Since  $n + 1 > |S|$ , path  $p$  contains at least one cycle and, thus, we know that  $p = s_0^p \dots s_i^p \dots s_j^p \dots s_n^p \dots$ , where  $s_i^p = s_j^p$  for at least two  $i, j$  such

that  $0 \leq i < j < n$ . Let  $p'$  be the path  $s_0^p \dots s_{i-1}^p c^\omega$ , where we  $c = s_i^p \dots s_j^p$ . Clearly,  $p' \neq p$  and  $p' \in \gamma(s)$ .

$$\begin{aligned}
& \llbracket [\phi_1 \mathbf{U}^{n+1} \phi_2] \rrbracket(p') \times \llbracket [\phi_1 \mathbf{U}^{n+1} \phi_2] \rrbracket(p) = \\
& \quad (\text{by Lemma 12.}) \\
& \llbracket [\phi_1 \mathbf{U}^j \phi_2] \rrbracket(p') \times \llbracket [\phi_1 \mathbf{U}^{n+1} \phi_2] \rrbracket(p) = \\
& \quad (\times \text{ idempotent implies } + \text{ distributes over } \times.) \\
& (\llbracket [\phi_1 \mathbf{U}^j \phi_2] \rrbracket(p') \times \llbracket [\phi_1 \mathbf{U}^n \phi_2] \rrbracket(p)) + (\llbracket [\phi_1 \mathbf{U}^j \phi_2] \rrbracket(p') \times t_p(s_0^p \dots s_n^p)) = \\
& \quad (\delta = \{t_p(s_0^p \dots s_l^p) \mid 0 \leq l < n\}, \delta' = \{t_{p'}(s_0^{p'} \dots s_l^{p'}) \mid 0 \leq l < j\}) \\
& (\sum_{t \in \delta'} t \times \sum_{t \in \delta} t) + (\sum_{t' \in \delta'} t' \times t_p(s_0^p \dots s_n^p)) = \\
& \quad (\text{By commutativity and associativity of } +. \text{ Lemma 13 where } t_0 = t'_0) \\
& (\sum_{t \in (\delta' / \{t'_0\}} t \times \sum_{t \in (\delta / \{t_0\})} t) + (\sum_{t \in (\delta' / \{t'_0\}} t \times t_p(s_0^p \dots s_n^p))) + \\
& \quad ((t'_0 \times t_0) + (t'_0 \times t_p(s_0^p \dots s_n^p))) = \\
& \quad (\times \text{ is idempotent (thus } t'_0 \times t_0 = t'_0) \text{ and absorption law. )} \\
& (\sum_{t \in (\delta' / \{t'_0\}} t_{p'} \times \sum_{t \in (\delta / \{t_0\})} t) + (\sum_{t' \in (\delta' / \{t'_0\}} t' \times t_p(s_0^p \dots s_n^p))) + (t'_0 \times t_0) = \\
& \quad (\text{By commutativity and associativity of } +.) \\
& (\sum_{t \in \delta'} t \times \sum_{t \in \delta} t) + (\sum_{t' \in (\delta' / \{t'_0\})} t' \times t_p(s_0^p \dots s_n^p))
\end{aligned}$$

Since this holds for every  $t'_0 \in \delta'$  and  $\delta'$  is finite, we can repeat this reasoning and conclude that  $(\llbracket [\phi_1 \mathbf{U}^j \phi_2] \rrbracket(p') \times \llbracket [\phi_1 \mathbf{U}^n \phi_2] \rrbracket(p)) + (\llbracket [\phi_1 \mathbf{U}^j \phi_2] \rrbracket(p') \times t_p(s_0^p \dots s_n^p))$  is just  $\llbracket [\phi_1 \mathbf{U}^j \phi_2] \rrbracket(p') \times \llbracket [\phi_1 \mathbf{U}^n \phi_2] \rrbracket(p)$ .

Hence,  $\llbracket [\phi_1 \mathbf{U}^{n+1} \phi_2] \rrbracket(p') \times \llbracket [\phi_1 \mathbf{U}^{n+1} \phi_2] \rrbracket(p) = \llbracket [\phi_1 \mathbf{U}^n \phi_2] \rrbracket(p') \times \llbracket [\phi_1 \mathbf{U}^n \phi_2] \rrbracket(p)$ .

It is fairly easy to show by induction on  $\gamma(s)$  and using the above result that  $\llbracket \Pi[\phi_1 \mathbf{U}^{n+1} \phi_2] \rrbracket$  is  $\llbracket \Pi[\phi_1 \mathbf{U}^n \phi_2] \rrbracket$ .

Finally, applying the first induction hypothesis, namely  $\llbracket \Pi[\phi_1 \mathbf{U}^n \phi_2] \rrbracket = \llbracket \Pi[\phi_1 \mathbf{U}^{|S|} \phi_2] \rrbracket$  we arrive to the desired result.  $\square$

**Corollary 15** *If  $C$  is distributive,  $S$  is finite and  $T$  total, any c-CTL formula  $\phi$  can be computed by using at most  $|S|$  iterations provided that any subformula of  $\phi$  has been already computed.*

Observe that the fixpoint formula corresponding to any c-CTL formula  $\phi$ , i.e. the one induced by the path semantics of c-CTL, is a formula in which no variable appears free under the scope of more than one fixpoint operator. In other words, in every such fixpoint formula  $\mu z. \phi_1 + \kappa \mathbf{X}z$ ,  $\nu z. \phi_1 \times \kappa \mathbf{X}z$ ,  $\mu z. \phi_2 + (\phi_1 \times \kappa \mathbf{X}z)$  or  $\nu z. \phi_2 \times (\phi_1 + \kappa \mathbf{X}z)$  subformulae  $\phi_1, \phi_2$  are closed. This

implies that we can compute the fixpoints inside a formula starting with the innermost ones and ending with the outermost ones as sketched in the algorithm below.

---

**Algorithm** *evalCTL*( $M, \phi$ )

**Input:** A transition system  $M$ , where  $C$  is distributive, a  $c$ - $L_\mu$  formula  $\phi$  corresponding to a  $c$ -CTL formula and an environment  $e$ .

**Output:** A valuation as an array of  $|S|$  elements.

**switch**  $\phi$  **do**

**case**  $\mu z. \phi_2 + (\phi_1 \times \kappa \mathbf{X}z)$

$v := \mathbf{0}$ ;  $v_1 := \text{evalCTL}(M, \phi_1, e)$ ;  $v_2 := M, \text{evalCTL}, e(\phi_2)$ ;

**repeat**  $v' := v$ ;  $v := \text{evalCTL}(M, v_1 + (v_2 \times \kappa \mathbf{X}z), e[v'/z])$  **until**

$v' = v$ ;

**return**  $v$ ;

**case**  $\nu z. \phi_2 \times (\phi_1 + \kappa \mathbf{X}z)$

    /\* similar as above \*/

**otherwise**  $\text{eval}(M, \phi, e)$ ;

**end**

---

Let  $t$  denote the time complexity of any  $c$ -semiring function, including  $+$  and  $\times$ .  $T$  depends on the actual  $c$ -semiring instance and is not necessarily  $O(1)$ . Take for example, the power set of a  $c$ -semiring which domain is infinite. The result is a  $c$ -semiring where elements of the domain are possibly infinite sets. Storing and manipulating such elements might be unfeasible.

An example of a worst-case  $c$ -CTL formula is  $\phi = \kappa[\phi_1 \mathbf{U}p]$ , where  $\phi_1$  has the same form or is an atomic proposition. The corresponding fixpoint formula is  $\mu z. p + (\psi \times \kappa \mathbf{X}z)$ , where  $\psi$  is the corresponding fixpoint formula of  $\phi_1$ . Observe that this formula has  $O(|\phi|)$  fixpoint subformulae, where  $|\phi|$  denotes the length of  $\phi$ . The time required to compute  $\phi$  is the time required to compute  $\phi_1$  plus  $|S|$  times the time required to perform an iteration which is  $O(|S| \cdot t)$  (time to compute  $+$  and  $\times$  for each state) plus  $O(|S|^2 \cdot t)$  (time to compute  $\kappa \mathbf{X}$  for each state). The resulting time complexity is  $O(|\phi| \cdot |S|^3 \cdot t)$ .

### 4.3 Model checking for fragments of $c$ -CTL.

We now concentrate in the general case, where  $C$  is not necessarily a distributive  $c$ -semiring. We restrict to the path semantics of formula in  $c$ - $\Sigma$ CTL, the fragment of  $c$ -CTL that has  $\Sigma$  as unique path quantifier.

**Model Checking**  $\Sigma[\phi_1 \mathbf{U} \phi_2]$ . We start showing that  $\Sigma[\phi_1 \mathbf{U} \phi_2]$  can be computed in  $O(|S|^3 \cdot t)$  time by reducing the evaluation of the formula to the *algebraic path problem* [24].

The algebraic path problem consists on performing a special unary operation, called *closure* over a square matrix, whose matrix are elements of a semiring.

The graph approach casts the problem as the computation of the addition of all weighted paths between each pair of nodes, where the weight of a path is the product of the weights of the edges of the path. Let  $C = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$  be a semiring and  $G = \langle U, w \rangle$  be a *weighted graph*, where  $U$  is a finite set of nodes and  $w : U \times U \rightarrow A$  is a function assigning weights (semiring values) to edges. The weight of a path  $p = v_0..v_n$  is defined as  $w(p) = w(v_0, v_1) \times \dots \times w(v_{n-1}, v_n)$ . The algebraic path problem is then reduced to the computation of  $d(u, v) = \sum_{p \in \gamma(u, v)} w(p)$  for each  $u, v \in U$ , where  $\gamma(u, v)$  denotes the set of all paths from  $u$  to  $v$ . General algorithms work for *closed* semirings, which are semirings with a closure, i.e. a unary operator  $*$  :  $A \rightarrow A$  such that  $\forall a \in A : a^* = \mathbf{1} + (a \times a^*) = \mathbf{1} + (a^* \times a)$ . It is easy to see that c-semirings are a specific case of closed semirings by letting  $a^*$  be  $\mathbf{1}$  for all  $a \in A$ . The running time of all known algorithms is  $O(|U|^3 \cdot t)$  [11,24], where  $t$  is the time necessary to perform the semiring operations.

Given a transition system  $M = \langle S, T, C, P, I \rangle$  and a formula  $\phi \equiv \sum[\phi_1 \mathbf{U} \phi_2]$  we construct a weighted graph  $G = \langle U, w \rangle$  such that:  $U = S \cup \{goal\}$ , where  $goal \notin S$ ,  $w(s, s')$  is  $\llbracket \phi_1 \rrbracket(s)$  iff  $(s, s') \in T$  and  $\mathbf{0}$  otherwise, and  $w(s, goal) = \llbracket \phi_2 \rrbracket(s)$  for all  $s \in S$ . The intuitive idea is that each path  $s..s'goal$  corresponds to the finite prefixes  $s..s'$  of the paths starting at  $s$  and passing through  $s'$ , which are considered in the computation of  $\llbracket \sum[\phi_1 \mathbf{U} \phi_2] \rrbracket(s)$ . Thus computing  $\llbracket \sum[\phi_1 \mathbf{U} \phi_2] \rrbracket(s)$  amounts to computing  $d(s, goal)$ . This is a special case of the algebraic path problem, namely the *all-sources single-target generalized shortest path*.

**Theorem 16** *Evaluating  $\sum[\phi_1 \mathbf{U} \phi_2]$  can be done in  $O(|S|^3 \cdot t)$  via the algebraic path problem, provided that  $\phi_1, \phi_2$  have been already computed.*

**PROOF.** By definition  $\llbracket \sum[\phi_1 \mathbf{U} \phi_2] \rrbracket(s)$  is  $\sum_{p \in \gamma(s)} \llbracket \phi_1 \mathbf{U} \phi_2 \rrbracket(p)$ . Let  $\gamma^f(s)$  denote the set of all finite paths starting at  $s$ . Using associativity and idempotency of  $+$  it is easy to show that  $\sum_{p \in \gamma(s)} \llbracket \phi_1 \mathbf{U} \phi_2 \rrbracket(p)$  is equivalent to  $\sum_{p \in \gamma^f(s)} \llbracket \phi_1 \rrbracket(s_0^p) \times \dots \times \llbracket \phi_1 \rrbracket(s_{|p|-2}^p) \llbracket \phi_2 \rrbracket(s_{|p|-1}^p)$  which is exactly  $\sum_{p \in \gamma(s, goal)} w(s_0^p, s_1^p) \times \dots \times w(s_{|p|-1}^p, goal)$  by the construction of the weighted graph, which can be done in  $O(|S|^2)$  time.  $\square$

**Model checking  $\sum[\mathbf{G}\phi]$ .** In the following we abbreviate  $a \times a \times \dots$  with  $a^\omega$ . In many cases  $a^\omega$  can be easily computed. For instance, if  $\times$  is idempotent then  $a^\omega$  is just  $a$ , in other cases like in the probabilistic or optimization c-semirings,  $a^\omega$  is just  $\mathbf{0}$  unless  $a$  is  $\mathbf{1}$ . If the c-semiring is the Cartesian product of  $n$  c-semirings then  $a^\omega = (a_0, \dots, a_n)^\omega$  is  $(a_0^\omega, \dots, a_n^\omega)$  for all  $a \in A$ . In the case of the Hoare Powerset,  $a^\omega = \{a_0, a_2, \dots\}^\omega$  is  $\{a_0^\omega, a_1^\omega, \dots\}$ . In the following we assume that this operation can be done in time  $t$ .

We reduce the evaluation of  $\sum[\mathbf{G}\phi]$  to the algebraic path problem again. The rough idea is that evaluating  $\sum[\mathbf{G}\phi]$  in a state  $s$  amounts to computing the

addition of the weights of all infinite paths. Actually, one has to consider only paths  $pq^\omega$ , where  $p$  is acyclic and  $q$  is a *simple cycle*, i.e. a cycle not containing a cycle. Observe that this is correct since, any other infinite path  $p'$  necessarily contains a path  $pq^\omega$  (meaning that  $pq^\omega$  is a path in the underlying graph of  $p'$ ) starting from  $s$  and, due to monotonicity of the multiplicative operation the weight of  $p$  must be necessarily higher or equal than the weight of  $pq^\omega$ . This is part of the proof of Theorem 17.

Given a transition system  $M = \langle S, T, C, P, I \rangle$  and a formula  $\Sigma[\mathbf{G}\phi]$  we construct a weighted graph  $G = \langle U, w \rangle$  as follows: each state and of the transition system is a node. For each simple cycle  $c$  in the transition system we create two nodes  $v_c, v'_c$ . The weight function is defined such that  $w(s, s') = \llbracket \phi \rrbracket(s)$  if  $(s, s') \in T$ ,  $w(s, v_c) = \llbracket \phi \rrbracket(s)$  if  $s \in c$  and  $\mathbf{0}$  otherwise, and  $w(v_c, v_c) = (\prod_{s \in c} \llbracket \phi \rrbracket(s))^\omega$ , i.e. the infinite power of the product of the evaluation of  $\phi$  in every state cycle  $c$ . The evaluation of  $\llbracket \mathbf{G}\phi \rrbracket(s)$  is thus reduced to the computation of  $\sum_{c \in \mathcal{C}(G)} d(s, v'_c)$ , where  $\mathcal{C}(G)$  denotes the set of all simple cycles of  $G$ , which number might be exponential in the size of the graph.

**Theorem 17** *Evaluating  $\Sigma[\mathbf{G}\phi]$  can be done in  $O(c^{|S|} + |S|^3 \cdot t)$  for some constant  $c > 1$  via the algebraic path problem, provided that  $\phi$  has been already computed.*

**PROOF.** We first show that  $\sum_{p \in \gamma(s)} \llbracket \mathbf{G}\phi \rrbracket(p)$  is equal to  $\sum_{p \in \gamma^-(s)} \llbracket \mathbf{G}\phi \rrbracket(p)$ , where  $\gamma^-(s)$  is the set of all maximal paths starting at  $s$  that can be written as  $qc^\omega$ , where  $c$  is a simple cycle and  $q$  is acyclic. Briefly, the idea is the following: for each path  $p$  in  $\gamma(s) \setminus \gamma^-(s)$  there is a path  $p'$  in  $\gamma^-(s)$  such that  $\llbracket \mathbf{G}\phi \rrbracket(p) \leq_s \llbracket \mathbf{G}\phi \rrbracket(p')$ , and hence  $\llbracket \mathbf{G}\phi \rrbracket(p') + \llbracket \mathbf{G}\phi \rrbracket(p)$  is just  $\llbracket \mathbf{G}\phi \rrbracket(p')$ . This should be clear because if  $p = qc'q'c^\omega$ , where  $c'$  is a cycle we can construct  $p' = qq'c^\omega$ . By associativity of  $\times$ ,  $\llbracket \mathbf{G}\phi \rrbracket(p') = \llbracket \mathbf{G}\phi \rrbracket(p) \times \llbracket \mathbf{G}\phi \rrbracket(c')$ , which is clearly smaller or equal than  $\llbracket \mathbf{G}\phi \rrbracket(p)$ , by monotony of  $\times$ . The same can be done for a path  $p = q(c_1cc_2)^\omega$ , where  $c$  is a cycle. Now we see that each path in  $G$  from  $s$  to a node  $v'_c$  corresponds to a path in  $\gamma^-(s)$ . Thus,  $\sum_{p \in \gamma^-(s)} \llbracket \mathbf{G}\phi \rrbracket(p)$  is  $\sum_{c \in \mathcal{C}(G)} d(s, v'_c)$ .  $\square$

**Model Checking  $\Sigma[\phi_1 \mathbf{R} \phi_2]$ .** We show that the problem is decidable and give an upper bound on the complexity. The rough idea is that in order to evaluate  $\Sigma[\phi_1 \mathbf{R} \phi_2]$  in a state  $s$  one has to examine finitely many maximal paths only, among those starting at  $s$ . In addition, from each maximal path we have to take only a finite prefix into account. The resulting naive algorithm consists in enumerating all those finite prefixes  $p'$  and computing the addition of  $[\phi_1 \mathbf{R} \phi_2](p')$ .

**Theorem 18** *Evaluating  $\Sigma[\phi_1 \mathbf{R} \phi_2]$  can be done in  $O(c^{|S|} \cdot |S| \cdot t)$  for some constant  $c$ , provided that  $\phi_1, \phi_2$  have been already computed.*

**PROOF.** As in the proof of theorem 17 one can show that  $\sum_{p \in \gamma(s)} \llbracket \phi_1 \mathbf{R} \phi_2 \rrbracket(p)$  is the same as  $\sum_{p \in \gamma^-(s)} \llbracket \phi_1 \mathbf{R} \phi_2 \rrbracket(p)$ . For a path  $p = qc^\omega \in \gamma^-(s)$  one can show that  $\llbracket \phi_1 \mathbf{R} \phi_2 \rrbracket(p)$  is just  $\llbracket \phi_1 \mathbf{R} \phi_2 \rrbracket(q) \times (\llbracket \phi_1 \mathbf{R} \phi_2 \rrbracket(qc))^\omega$ , which can be computed in  $O(|qc|^2 \cdot t)$  time, where  $|qc|$  can be at most  $|S|$ . Because the number of paths in  $\gamma^-(s)$  can be exponential in the number of states the complexity of evaluating  $\llbracket \sum \phi_1 \mathbf{R} \phi_2 \rrbracket(s)$  is at most  $O(c^{|S|} \cdot |S| \cdot t)$ , for some constant  $c > 1$ .  $\square$

## 5 Equivalence relations and preorders.

Equivalence and preorder relations between transition systems are fundamental issues to support the tractability of model checking problems. Methods to reduce the complexity of the problem base on such relations to guarantee correctness. For example, since most temporal logics do not distinguish between bisimilar systems, checking a transition system can be done by analyzing a bisimilar system, possibly easier to handle.

As advanced in the introduction we will provide boolean notions of equivalence and preorder relations, and not quantitative notions, since it is not clear how to define a meaningful notion of distance in c-semirings. Note that the values the domain of a c-semiring are partially ordered: some values are not comparable and it is not clear which the distance between such states should be.

We define a natural extension of bisimulation to our approach which results in a relation  $\sim_B$  on states and we shall see that our logics distinguish bisimilar states in general.

**Definition 19** A binary relation  $\sim_B$  on  $S$  is a *bisimulation relation* whenever for all  $s, s' \in S$ ,  $s \sim_B s'$  implies:

- For all  $p \in AP$  we have  $I(p)(s) = I(p)(s')$ ;
- $\forall (s, s_1) \in T : \exists (s', s'_1) \in T \mid s_1 \sim_B s'_1$ ;
- $\forall (s', s'_1) \in T : \exists (s, s_1) \in T \mid s'_1 \sim_B s_1$ .

**Example 20** Consider Example 6. Suppose we have two additional states  $s'_0, s'_1$  and one transition  $(s'_0, s'_1)$ . Assume that  $p$  represents the price to pay for the usage of the resource which is 1 in all states. It is easy to find a bisimulation relation  $\sim_B$  such that  $s_0 \sim_B s'_0$ . Consider now, we want to cumulate the prices of the usage of the resource in all next states. The corresponding formula is  $\llbracket \mathbf{X}p \rrbracket$ . However,  $\llbracket \mathbf{X}p \rrbracket(s_0) = 2$  and  $\llbracket \mathbf{X}p \rrbracket(s'_0) = 1$ .

Hence, our logics distinguishes bisimilar states because of the non-idempotency of  $\times$  in general. For example, we can count the number of branches and hence clearly distinguish bisimilar states.

We now give a theorem that identifies some cases in which the bisimulation

$\sim_B$  is an equivalence relation for our logic: if the c-semiring is distributive or we consider a fragment of c-CTL that does not include  $\prod$  as path quantifier. We shall call this fragment c- $\overline{\prod}$ CTL.

**Theorem 21** *If  $\sim_B$  is a bisimulation relation,  $\phi \in c\text{-}\overline{\prod}\text{CTL}$  or  $\phi \in c\text{-}L_\mu$  and  $C$  is distributive then for all  $s, s' \in S$ ,  $s \sim_B s'$  implies  $\llbracket \phi \rrbracket(s) = \llbracket \phi \rrbracket(s')$ .*

**PROOF.** We will prove the theorem by induction on the subformulae of  $\phi$ . Suppose that  $C$  is distributive. First, it is easy to see that the theorem holds for the trivial cases  $\mathbf{0}, \mathbf{1}, p$ . By applying induction it is also clear that the theorem holds for  $f(\phi_1, \dots, \phi_n)$ ,  $\phi_1 + \phi_2$  and  $\phi_1 \cdot \phi_2$ .

Let us now consider the case when  $\phi$  is  $\kappa \mathbf{X} \phi_1$ . Applying the induction hypothesis we have that  $\forall (s, s_1) \in T : \exists (s', s'_1) \in T \mid \llbracket \phi_1 \rrbracket(s_1) = \llbracket \phi_1 \rrbracket(s'_1)$  and  $\forall (s', s'_1) \in T : \exists (s, s_1) \in T \mid \llbracket \phi_1 \rrbracket(s'_1) = \llbracket \phi_1 \rrbracket(s_1)$ . Hence we have,  $\{\llbracket \phi_1 \rrbracket(s_1) \mid (s, s_1) \in T\}$  is equal to  $\{\llbracket \phi_1 \rrbracket(s'_1) \mid (s', s'_1) \in T\}$ . Since  $\times$  is idempotent, then  $\prod$  coincides with  $\overline{\prod}$ . Thus every quantifier  $\kappa$  is idempotent. Thus, for every quantifier  $\kappa$ , we have  $\llbracket \kappa \mathbf{X} \phi_1 \rrbracket(s) = \kappa \{\llbracket \phi_1 \rrbracket(s_1) \mid (s, s_1) \in T\} = \kappa \{\llbracket \phi_1 \rrbracket(s'_1) \mid (s', s'_1) \in T\} = \llbracket \kappa \mathbf{X} \phi_1 \rrbracket(s')$ .

If  $\phi$  is a fixpoint formula  $\mu z. \phi_1$  or  $\nu z. \phi_1$  observe that the set of fixpoints of  $\phi_1 \{v/z\} \{v \in V \mid \tau(v) = v\}$  is a subset of  $\{v \in V \mid \tau(v)(s) = v(s)\}$ . Applying induction on  $\phi_1$  we have that for any fixpoint  $v$ ,  $\phi_1$  evaluates to the same value in  $s$  and  $s'$ . We conclude that  $\llbracket \mu z. \phi_1 \rrbracket(s) = \llbracket \mu z. \phi_1 \rrbracket(s')$  and  $\llbracket \nu z. \phi_1 \rrbracket(s) = \llbracket \nu z. \phi_1 \rrbracket(s')$ .

Now assume that  $C$  is not distributive. For all cases but  $\kappa[\phi_1 \mathbf{T}_{\circlearrowleft} \phi_2]$  the proof is the same as above, where  $\kappa \in \{\sum, \prod\}$ . We hence concentrate in the case  $\phi = \kappa[\phi_1 \mathbf{T}_{\circlearrowleft} \phi_2]$ .

Applying the induction hypothesis it is easy to see that for each path  $p$  in  $\gamma(s)$  there is a bisimilar path  $p'$  in  $\gamma(s')$  such that  $\llbracket \phi_1 \mathbf{T}_{\circlearrowleft} \phi_2 \rrbracket(p)$  is equal to  $\llbracket \phi_1 \mathbf{T}_{\circlearrowleft} \phi_2 \rrbracket(p')$ , and vice versa. Thus  $\{\llbracket \phi_1 \mathbf{T}_{\circlearrowleft} \phi_2 \rrbracket(p) \mid p \in \gamma(s)\}$  equals  $\{\llbracket \phi_1 \mathbf{T}_{\circlearrowleft} \phi_2 \rrbracket(p) \mid p \in \gamma(s')\}$ . Hence, since  $\kappa$  is idempotent  $\llbracket \kappa[\phi_1 \mathbf{T}_{\circlearrowleft} \phi_2] \rrbracket(s) = \kappa \{\llbracket \phi_1 \mathbf{T}_{\circlearrowleft} \phi_2 \rrbracket(p) \mid p \in \gamma(s)\} = \kappa \{\llbracket \phi_1 \mathbf{T}_{\circlearrowleft} \phi_2 \rrbracket(p) \mid p \in \gamma(s')\} = \llbracket \kappa[\phi_1 \mathbf{T}_{\circlearrowleft} \phi_2] \rrbracket(s')$ .  $\square$

To further illustrate the properties of distributive c-semirings and the  $\prod$ -free of c-CTL we define a natural adaptation of the notion of simulation.

**Definition 22** A binary relation  $\prec$  on  $S$  is a *simulation relation* whenever for all  $s, s' \in S$ ,  $s \prec s'$  implies:

- $\forall p \in AP : I(p)(s) \leq_S I(p)(s')$ ;
- $\forall (s, s_1) \in T : \exists (s', s'_1) \in T \mid s_1 \prec s'_1$ .

**Theorem 23** *If  $\prec$  is a simulation relation,  $\phi \in c\text{-}\overline{\text{CTL}}$  or  $\phi \in c\text{-}L_\mu$  and  $C$  is distributive then for all  $s, s' \in S$ ,  $s \prec s'$  implies  $\llbracket \phi \rrbracket(s) \leq_s \llbracket \phi \rrbracket(s')$ .*

**PROOF.** The proof is very similar to the proof of Theorem 21, but one uses set containment instead of set equality and monotony of the operations. For example, for  $\phi = \kappa \mathbf{X} \phi_1$  with a distributive c-semiring we see that  $\{\llbracket \phi_1 \rrbracket(s_1) \mid (s, s_1) \in T\} \subseteq \{\llbracket \phi_1 \rrbracket(s'_1) \mid (s', s'_1) \in T\}$ . Now, by idempotency and monotony of  $\kappa$  we have  $\llbracket \kappa \mathbf{X} \phi_1 \rrbracket(s) = \kappa\{\llbracket \phi_1 \rrbracket(s_1) \mid (s, s_1) \in T\} \leq_S \kappa\{\llbracket \phi_1 \rrbracket(s'_1) \mid (s', s'_1) \in T\} = \llbracket \kappa \mathbf{X} \phi_1 \rrbracket(s')$ .  $\square$

## 6 Applications

Along the paper we have illustrated the concepts with a running example, where a system is composed by agents accessing resources with a certain QoS. We have shown different examples of quantitative properties regarding the QoS level of the usage. In this section we motivate the interest in our approach by showing additional applications different from scenarios involving QoS properties. Some of these examples are well-known applications that our framework captures.

**Boolean Model Checking.** Our approach can be specialized to traditional model checking by simply letting  $C$  be the boolean c-semiring. Other boolean approaches based on multi-valued logics are capture too. Multi-valued CTL ( $\chi$ CTL) [9] is defined over quasi-boolean algebras, which are finite distributive lattice with a negation operator. It is easy to see that given a quasi boolean algebra  $\langle A, \sqcap, \sqcup, \neg \rangle$ , the algebraic structure  $\langle A, \sqcup, \sqcap, \perp, \top \rangle$  with  $\perp$  and  $\top$  denoting the bottom and top elements of the lattice is a c-semiring where the multiplicative operation is idempotent.  $\chi$ CTL is interpreted over  $\chi$  Kripke structures, which are basically Kripke structures where each transition has an associated value which is taken into account when computing the semantics of next-state operators. Roughly, the value of the transition is multiplied. This can be achieved using  $c\text{-}L_\mu$  but not by  $c\text{-CTL}$ .

**Graph Problems.** As boolean model checking can be used to express some boolean graph properties, like reachability, our logics can be used to express graph properties involving costs. We show how to reduce the algebraic path problem (where the algebraic structure is a c-semiring) to  $c\text{-CTL}$ .

Let  $C$  be the c-semiring that models the costs and let  $G = \langle U, w \rangle$  be a weighted graph. We construct a transition system  $M = \langle S, T, C, P, I \rangle$  as follows:  $S = U \cup (U \times U)$ ,  $(u, v), ((u, v), v) \in U$  for all  $u, v \in S$  and  $P = \{cost \cup \bigcup_{u \in U} goal_u\}$ , where  $I(cost)(u) = \mathbf{1}$ ,  $I(cost)((u, v)) = w(u, v)$  and  $I(goal_u)(v)$  is  $\mathbf{1}$  if  $u = v$  and  $\mathbf{0}$  otherwise. The idea is that the transition system has node-states and transition-states. Only the latter have a cost associated by proposition  $cost$ .

Computing  $d(u, v)$  amounts to evaluating  $\llbracket \text{costUgoal}_v \rrbracket(u)$ . We omit the proof due to lack of space, but the intuitive idea is that one takes the addition for all paths starting at  $u$  of the addition of weight of all prefixes of the path. Note that prefixes not ending at  $v$  just receive value  $\mathbf{0}$ .

**Discounted model checking.** Roughly speaking, discounted model checking [12,13] weights events according to their distance to the present state, i.e. along a path  $s_0, s_1, \dots$  the evaluation of a formula in  $s_i$  is multiplied by  $c^i$ , where  $c$  is the a discounting factor between 0 and 1. DCTL [13], for instance, is a discounted version of CTL. In addition to  $\min$  and  $\max$ , it uses additional operators over  $[0, 1]$ , namely  $-$ ,  $\cdot$ ,  $+$  and  $+_c$ , which can be represented by functions in  $F$ . The temporal (discounted) operators of DCTL are  $\diamond_c$ ,  $\square_c$ ,  $\Delta_c$ , respectively correspond to the application of  $\min$ ,  $\max$  and  $\text{average}$  to the discounted values of the states in a path.

Discounted model checking of transition systems can be performed using  $c$ -semiring  $\mu$ -calculus as follows. The  $c$ -semiring is  $\langle [0, 1], \max, \min, 0, 1 \rangle$ . The semantics of DCTL for transition systems can be described with  $c$ - $L_\mu$  by using a direct translation of the fixpoint semantics of DCTL. For instance,  $\exists \diamond_c \phi \equiv \mu z. \phi + (\mathbf{0} +_c \sum \mathbf{X}z)$  and  $\forall \diamond_c \phi \equiv \mu z. \phi + (\mathbf{0} +_c \prod \mathbf{X}z)$ .

**Model Checking with Resources.** Reasoning about resource constraints [8] is another potential field of application of our framework. Consider the simple case in which the system consumes a certain amount of energy in every state. We might be interested in knowing whether power consumption does not exceed a certain value. If we use the  $c$ -semiring  $\langle \mathbb{R}^+, \min, \max, \infty, 0 \rangle$ , and let  $v(s)$  denote the power consumption at state  $s$ , then formula  $\sum \mathbf{G}v$  evaluated in a state  $s_0$  represents the maximal power consumption of optimal execution of the system starting at  $s_0$ .

## 7 Conclusions and Future Work

We have presented extensions of two well-known temporal specification formalism: CTL and the  $\mu$ -calculus. While the original logics are defined over the boolean domain, our extensions are defined over the domain of a  $c$ -semiring, an algebraic structure that captures various quantitative aspects like soft constraints or QoS attributes. Along the paper we have distinguished a specific kind of  $c$ -semirings, which are actually distributive lattices. We have also distinguished some fragments of our logics when discussing semantics, algorithms and equivalence relations.

We have first defined syntax and semantics of our logics over transition systems. We have given two semantics to  $c$ -CTL: The classical path semantics and the fixpoint semantics, which are equivalent in boolean model checking. Both semantics coincide for distributive  $c$ -semirings. In the general case, the

equivalence holds for a fragment only. For formulae of the fragment of  $c$ -CTL that uses only the additive operation as path quantifier (called  $c$ - $\Sigma$  CTL) the path and fixpoint semantics are related by the partial order of the  $c$ -semiring. More precisely, the fixpoint semantics return smaller values.

We have shown that model checking  $c$ - $L_\mu$  for distributive  $c$ -semirings and restricting to finite-state transition systems can be done by fixpoint iteration, establishing an upper bound on the complexity. This is an interesting result since it applies to instances of our framework that define logics over (possibly) infinite distributive lattices like boolean algebras [9] or the fuzzy  $c$ -semiring [12]. Focusing on  $c$ -CTL we have refined the complexity bound by showing that each fixpoint requires a number of iterations equal to the number of states in the underlying state transition graph.

For general  $c$ -semirings, two problems arise. First, since the path and fixpoint semantics differ, each one requires different algorithms. Second, fixpoint iteration is not guaranteed to terminate. We have thus concentrated on model checking the path semantics of formulae of  $c$ - $\Sigma$  CTL. We give upper bounds on the complexity of evaluating different formulae, in most cases by reducing the evaluation to the algebraic path problem, a general problem on graphs which cost formalism are semirings more general than  $c$ -semirings. In current work we are devising solutions for cases for other fragments of our logics.

We have defined a bisimulation relation for transition systems and shown that it is an equivalence relation for fragments of our logics only, including  $c$ - $\Sigma$  CTL. Similarly, only for  $c$ - $\Sigma$  CTL and other fragments one can define a simulation relation such that if one state is simulated by other then the evaluation of the formulae in each state are related by the partial order of the  $c$ -semiring. Not surprisingly, the bisimulation and simulation relations have the desired properties when one restricts to distributive  $c$ -semirings.

One of the main interesting avenues would be to define a logic for reasoning about structural, behavioral and QoS aspects of WAN applications. Graphs and graphs transformation systems [25] are a suitable formalism for such systems, representing issues such as concurrency, distribution and mobility. In such formalism the transition system can be seen as graph transition system, where states are graphs. Approaches to express and verify boolean properties of graph transitions systems already exist, e.g., [23,3] and can serve as inspiration for our purposes. In [16] we introduce a simple graph logic for QoS. The evaluation of a formula in that logic is a value of  $c$ -semiring, representing the QoS level of the formula and not just a boolean value expressing whether or not the formula holds. A first approach could be to include formulae of the spatial logic as propositions of  $c$ -CTL or  $c$ - $L_\mu$ , such that the  $c$ -semirings used in both the graph and temporal logic have the same domain. In our example, e.g., states might be represented by a network, where agents and resources are at different locations and the QoS level of a usage is partially given by the QoS of the connection from the agent's location to the resource's one.

## Acknowledgements

The authors wish to thank the anonymous referees for their fruitful criticism.

## References

- [1] Alur, R., C. Courcoubetis and D. Dill, *Model-checking for real-time systems*, in: *5th Annual IEEE Symposium on Logic in Computer Science (LICS'90)* (1990), pp. 414–425.
- [2] Baier, C. and E. Clarke, *The algebraic mu-calculus and mtbdd*s, in: *Proc. 5th Workshop on Logic, Language, Information and Computation, (WoLLIC'98)*, 1998, pp. 27–38.
- [3] Baldan, P., A. Corradini, B. König and B. König, *Verifying a behavioural logic for graph transformation systems*, in: *CoMeta'03*, ENTCS **104**, 2004, pp. 5–24.
- [4] Bistarelli, S., H. Fargier, U. Montanari, F. Rossi, T. Schiex and G. Verfaillie, *Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison*, CONSTRAINTS (1999), pp. 199–240.
- [5] Bistarelli, S., H. Fargier, U. Montanari, F. Rossi, T. Schiex and G. Verfaillie, *Semiring-based constraint logic programming: Syntax and semantics*, ACM Transactions on Programming Languages and Systems (TOPLAS) **23** (2001), pp. 1–29.
- [6] Bistarelli, S., U. Montanari and F. Rossi, *Semiring-based constraint satisfaction and optimization*, Journal of the ACM **44** (1997), pp. 201–236.
- [7] Bistarelli, S., U. Montanari and F. Rossi, *Soft constraint logic programming and generalized shortest path problems*, Journal of Heuristics **8** (2002), pp. 25–41.
- [8] Chakrabati, A., L. de Alfaro, T. A. Henzinger and M. Stoenlinga, *Resource interfaces*, in: *Third International Conference on Embedded Software (EMSOFT'02)*, number 2855 in Lecture Notes in Computer Science (2003), pp. 117–133.
- [9] Chechik, M., S. Easterbrook and A. Gurfinkel, *Multi-valued symbolic model-checking*, ACM Transactions on Software Engineering and Methodology (2003), to appear.
- [10] Clarke, E. M., O. Grumberg and D. A. Peled, “Model Checking,” MIT Press, 1999.
- [11] Cormen, T. H., C. E. Leiserson, R. L. Rivest and C. Stein, “Introduction to Algorithms,” MIT Press, 2001.
- [12] de Alfaro, L., *Quantitative verification and control via the mu-calculus*, in: *Proceedings of 14th International Conference on Concurrency Theory*, Lecture Notes in Computer Science **2761** (2003), pp. 102–126.

- [13] de Alfaro, L., M. Faella, T. A. Henzinger, R. Majumdar and M. Stoelinga, *Model checking discounted temporal properties*, in: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, Lecture Notes in Computer Science **2988** (2004), pp. 77–92.
- [14] Duan, Z., Z. Zhang and Y. T. Hou, *Service overlay networks: SLAs, QoS, and bandwidth provisioning*, IEEE/ACM Transactions on Networking (TON) **11** (2003), pp. 870–883.
- [15] Emerson, E. A., A. K. Mok, A. P. Sistla and J. Srinivasan, *Quantitative temporal reasoning*, in: *Proceedings of the 2nd International Workshop on Computer Aided Verification* (1991), pp. 136–145.
- [16] Ferrari, G. and A. Lluch-Lafuente, *A logic for graphs with QoS*, in: *1st Workshop on Views On Designing Complex Architectures (VODCA)*, Electronic Notes in Theoretical Computer Science (2004), to appear.
- [17] Hansson, A. and B. Jonsson, *A logic for reasoning about time and probability*, Formal Aspects of Computing **12** (1994), pp. 512–535.
- [18] Huth, M. and M. Z. Kwiatkowska, *Quantitative analysis and model checking*, in: *Logic in Computer Science*, 1997, pp. 111–122.
- [19] Lluch-Lafuente, A. and U. Montanari, *Quantitative mu-calculus and CTL based on constraint semirings*, in: *2nd Workshop on Quantitative Aspects of Programming Languages*, Electronic Notes in Theoretical Computer Science **112** (2005), pp. 37–59.
- [20] Lunnon, W. F., *The  $iu$  function: the size of a free distributive lattice*, Combinatorial Mathematics and Its application (1971), pp. 173–181.
- [21] McIver, A. and C. Morgan, *Games, probability and the quantitative  $\mu$ -calculus*, in: *Logic Programming, Artificial Intelligence, and Reasoning (LPAR'02)*, number 2514 in Lecture Notes in Computer Science (2002), pp. 292–310.
- [22] Nicola, R. D., G. Ferrari, U. Montanari, R. Pugliese and E. Tuosto, *A formal basis for reasoning on programmable QoS*, in: N. Dershowitz, editor, *International Symposium on Verification (Theory and Practice)*, number 2772 in Lecture Notes in Computer Science (2003), pp. 436–479.
- [23] Rensink, A., *Towards model checking graph grammars*, in: *3rd Workshop on Automated Verification of Critical Systems*, Tech. Report DSSE-TR-2003, 2003, pp. 150–160.
- [24] Rote, A., *A systolic array algorithm for the algebraic path problem (shortest path; matrix inversion)*, Computing (1985), pp. 191–219.
- [25] Rozenberg, G., editor, “Handbook of graph grammars and computing by graph transformations,” World Scientific, 1997, 313–400 pp.
- [26] Seidl, H., *A modal mu-calculus for durational transition systems*, in: *LICS'96*, 1996, pp. 128–137.
- [27] Tarski, A., *A lattice-theoretical fixpoint theorem and its applications*, Pacific Journal of Mathematics (1955).